SHARP

TASCHENCOMPUTER

MODELL PC-1212

BEDIENUNGSANLEITUNG

12. KORREKTUR UND ÄNDERUNG (EDITIEREN) 24
12.1 Markierung der Korrekturstelle 25
12.2 Überschreiben 25
12.3 Löschen 25
12.4 Einfügen neuer Zeichen 25
12.5 Abruf des Ergebnisses nach der Korrektur 25
13. RANGORDNUNG DER OPERATOREN 27
14. SPEICHERORGANISATION 28
14.1 Eingaberegister 28
14.2 Rechenregister X 28
14.3 Pufferregister 28
14.4 Konstantenregister 30
14.5 Funktionsregister 30
14.6 Programmregister 30
14.7 RESERVE-Register 30

V PROGRAMMIEREN IN BASIC 31
1. AUFBAU DER SPRACHE BASIC 31
2. PROGRAMMAUFBAU 31
3. PROGRAMMEINGABE 34
4. ÜBERPRÜFUNG DER PROGRAMME 35
4.1 Überprüfung beim Eintasten 35
4.2 Überprüfen nach dem Einlesen in den Programmspeicher 35
4.3 Überprüfung nach Eingabe des gesamten Programms 35
5. PROGRAMMAUSFÜHRUNG 36
6. PROGRAMMKORREKTUR 37
6.1 Korrektur von Programmteilen 37
6.2 Einfügen von Programmzeilen 38
6.3 Löschen von Programmzeilen 39
7. FEHLERSUCHE 39
8. ORGANISATION DES PROGRAMMSPEICHERS 40
8.1 Eingaberegister 40
8.2 Programmregister 40
8.3 Anordnung der Programmzeilen im Programmregister 41
9. PROGRAMMNAMEN (LABEL) 42
9.1 Benennen des Programms 42
9.2 Kontrolle des benannten Programms 43
9.3 Abrufen des benannten Programms 43
9.4 Texte als Programmnamen 44

VI ZUORDNUNG VON AUSDRÜCKEN ZU TASTEN 45
1. EINGABE DER RESERVE-AUSDRÜCKE 45
2. ÜBERPRÜFUNG DER RESERVE-AUSDRÜCKE 46
3. ABRUF DER RESERVE-AUSDRÜCKE 47
4. KORREKTUR DER RESERVE-AUSDRÜCKE 49
5. ORGANISATION DES RESERVE-REGISTERS 49

VII VARIABLEN 50
1. VARIABLENTYPEN 50
1.1 Zahlenvariablen 50
1.2 Zeichenvariablen (Stringvariablen) 50
2. ORGANISATION DER VARIABLENSPEICHER 50

2.1 Der Basisspeicher . . . 50
2.2 Der flexible Speicher . . . 51
3. EINGABE VON VARIABLEN . . . 52
3.1 Manuelle Eingabe . . . 52
3.2 Programmierte Variableneingabe . . . 52
4. ABRUF VON VARIABLEN . . . 53
5. INDIREKTE ADRESSIERUNG . . . 54

VIII PROGRAMMANWEISUNGEN . . . 56
1. EINGABEANWEISUNGEN . . . 56
1.1 Wertzuweisung LET . . . 56
1.2 Eingabe mit der INPUT-Anweisung . . . 57
1.21 Allgemeine Formen der INPUT-Anweisung . . . 57
1.22 Überspringen einer Zeile . . . 58
1.3 Eingabe mit der AREAD-Anweisung . . . 59
2. AUSGABEANWEISUNGEN . . . 60
2.1 Die PRINT-Anweisung . . . 60
2.2 Die PAUSE-Anweisung . . . 63
2.3 Die USING-Anweisung . . . 63
3. STEUERANWEISUNGEN . . . 66
3.1 Unbedingte Sprunganweisung GOTO . . . 66
3.2 Programmverzweigungsanweisung IF . . . 67
3.3 Unterprogrammtechnik GOSUB RETURN . . . 69
3.4 Schleifenanweisungen FOR TO STEP NEXT . . . 71
3.5 Programmunterbrechung STOP . . . 75
3.6 Programmende END . . . 75
3.7 Löschanweisung CLEAR . . . 75
3.8 Winkeleinheit DEGREE RADIAN GRAD . . . 76
3.9 Akustisches Signal BEEP . . . 76
4. DIE KOMMENTARVEREINBARUNG REM . . . 76
5. KOMMANDOANWEISUNGEN . . . 76
5.1 Programmstart mit RUN . . . 76
5.2 Programmüberprüfung mit DEBUG . . . 77
5.3 Programmfortführung mit CONT . . . 77
5.4 Auflisten der Programme mit LIST . . . 78
5.5 Löschen mit NEW . . . 79
5.51 NEW in den Betriebsarten DEF, RUN und PRO . . . 79
5.52 NEW in der Betriebsart RESERVE . . . 79
5.6 Abfragen des verfügbaren Speicherplatzes mit MEM . . . 79
6. KOMMANDOTASTEN . . . 80
6.1 Löschtasten ON/CA CL . . . 80
6.2 Manuelle Programmunterbrechung ON/BREAK . . . 80
6.3 Eingabetaste ENTER . . . 80

IX FEHLERBEHANDLUNG . . . 81
1. FEHLER BEIM MANUELLEN RECHNEN . . . 81
2. PROGRAMMFEHLER . . . 81
3. FEHLERCODE UND FEHLERURSACHEN . . . 82

X DATENSPEICHERUNG AUF MAGNETBAND . . . 83
1. ANFORDERUNGEN AN REKORDER ODER TONBANDMASCHINE . . . 83
2. ANSCHLUSS DES RECHNERS AN DAS INTERFACE . . . 83

2.1 Einsetzen der Batterien in das Interface 83
2.2 Verbinden des Computers mit dem Interface 84
3. ANSCHLUSS DES INTERFACE AN DAS BANDGERÄT 85
4. ANWEISUNGEN FÜR DEN DIALOG RECHNER-BAND 85
4.1 Speichern von Programmen und RESERVE-Ausdrücken CSAVE 86
4.11 Programmspeicherung 87
4.12 Speichern von RESERVE-Ausdrücken. 87
4.2 Laden von Programmen und RESERVE-Ausdrücken CLOAD 87
4.21 Laden von Programmen 87
4.22 Laden von RESERVE-Ausdrücken 87
4.3 Vergleich der Bandinformation mit der Rechnerinformation CLOAD?. 88
4.31 Überprüfen von Programmen. 88
4.32 Überprüfen von RESERVE-Ausdrücken 88
4.4 Rechnergesteuerte Anforderung von Bandprogrammen CHAIN 88
4.5 Speichern von Daten PRINT# 90
4.6 Laden von Daten INPUT#. 91
5. VORBEREITUNG DER BANDARBEIT 91
6. SPEICHERN VON INFORMATIONEN AUF BAND 92
6.1 Manuelles Überspielen 92
6.2 Programmgesteuertes Überspielen. 92
7. LADEN DES RECHNERS VON BAND 93
7.1 Manuelles Laden 93
7.2 Programmgesteuertes Laden 93
7.3 Fehlermeldung beim Laden. 93
8. VERGLEICH VON BANDBLOCK MIT RECHNERBLOCK. 94
XI DIE TASTENFUNKTIONEN 95
XII LISTE DER FUNKTIONEN UND ANWEISUNGEN. 99
XIII TECHNISCHE DATEN 104

Durch äußere Störeinflüße mag es manchmal vorkommen, daß der PC-1212 blockiert und sich auch durch die CA/BREAK/ON-Tasto nicht mehr bedienen läßt. In einen solchen Fall betätigen Sie den "ALL RESET"-Schalter auf der Rückseite des Gerätes. Durch diese Maßnahme werden auch die Programm-und Datenspeicher gelöscht und müssen deshalb wieder neu geladen werden.
"ALL RESET", Siehe auch Seite 6.

I BETRIEBSVORBEREITUNG UND BEDIENUNG

1. EINLEITUNG

Der BASIC-programmierbare Computer PC-1212 ist ein leistungsfähiger wissenschaftlicher Rechner modernster Konzeption. Durch integrierte Elektronik konnte die Kapazität eines großen Tischgerätes im Taschenformat untergebracht werden. Für die große und kontrastreiche Darstellung von Zahlen und Zeichen sorgt eine 24-stellige Flüssigkristallanzeige. Erstmalig konnte bei einem Taschenrechner die höhere Programmiersprache BASIC verwendet werden, eine Sprache, die eine äußerst einfache Programmierung komplexer Probleme ermöglicht.

Allerdings muß man sich erst von einem gewöhnlichen Taschenrechner auf den Computer umstellen. So findet man die gewohnten mathematischen Funktionen nicht vielen Tasten mit Doppel- und Dreifachfunktionen zugeordnet, sondern man schreibt sie einfach mit der Zeichentastatur, die von der Schreibmaschine bekannt ist. Statt der Sinustaste drückt man also die Folge S I N. Weiter sind einige Zeichen unterschiedlich: Der Multiplikationsbefehl lautet * anstelle von x, der Divisionsbefehl / anstelle von ÷ und die allgemeine Exponentialfunktion verwendet das Zeichen ^ anstelle der Taste y^x. Der größte Unterschied liegt beim Gebrauch des Gleichheitszeichens. Mit = wird nicht wie beim Taschenrechner der Befehl zum Berechnen eines Ausdrucks gegeben — dafür hat der Computer die Taste ENTER — sondern die Zuordnung einer Größe zu einer Variablen ausgedrückt. Den formalmathematisch unlogischen BASIC-Satz A = A + 1 muß man daher wie folgt lesen: Der Variablen A wird der Zahlenwert A + 1 zugeordnet; hatte A ursprünglich den Wert 6 wird mit dem BASIC-Satz der Wert 7 zugeordnet. An diese Besonderheiten gewöhnt man sich aber sehr schnell, vorallem da versucht wurde, die BASIC-Sprache mit möglichst wenig fremdartigen Fachausdrücken einzuführen.

Damit Sie alle Möglichkeiten dieses vielseitigen BASIC-Computers kennenlernen, haben wir die mit vielen Beispielen und Rechentips versehene Bedienungsanleitung zusammengestellt. Lesen Sie bitte die folgenden Seiten! Der Zeitaufwand dafür bringt Ihnen beim Gebrauch des Rechners wiederum viel Zeitersparnis.

2. BETRIEBSHINWEISE

(1) Die Flüssigkristall-Anzeige dieses Rechners ist in Spezialglas eingebettet und kann daher bei Gewalteinwirkung zerbrechen. Bitte behandeln Sie deshalb den Rechner mit Sorgfalt. Stecken Sie das Gerät nicht in die Hosentasche, es kann beim Sitzen beschädigt werden. Beim Transport immer die schützende Hülle verwenden.

(2) Schützen Sie den Rechner vor Staub, Feuchte und allzugroßen Temperaturschwankungen.

(3) Zur Reinigung dient ein weiches, trockenes Tuch. Keine Reinigungs- oder Lösungsmittel verwenden.

(4) Wechseln Sie erschöpfte Batterien bitte sofort aus. Leere Batterien können undicht werden und das Gerät zerstören. Werfen Sie die alten Batterien auf keinen Fall ins Feuer; die Sprengwirkung ist verheerend.

(5) Auch bei sehr langem Nichtgebrauch sollten die Batterien außerhalb des Rechners aufbewahrt werden.

(6) Um statische Aufladung zu vermeiden, berührt man vor dem Öffnen des Geräts einen geerdeten Körper, etwa eine Wasserleitung.

(7) Wartungssarbeiten sollte allein die autorisierte Kundendienststelle durchführen.

(8) Dem Rechner ist ein selbstklebendes Namensschildchen beigepackt, das man an der Rückseite des Rechners befestigen kann.

3. STROMVERSORGUNG

ABSCHALTAUTOMATIK: Vier kleine Knopfzellen reichen als Stromquelle für etwa 300 Stunden. Um diese hohe Lebensdauer voll auszunützen, ist eine Abschaltautomatik eingebaut, die den Rechner nach 5 bis 8 Minuten ausschaltet, sobald nicht mehr gerechnet wird. Da der Rechner Permanentspeicher besitzt, werden durch die Abschaltung weder Daten noch Programme gelöscht. Sie stehen beim Einschalten sofort wieder zur Verfügung.

BATTERIEZUSTANDSANZEIGE: Rechts oben in der Anzeige befindet sich ein kleiner Kontrollpunkt, die Batteriezustandsanzeige. Wenn dieser Punkt nicht mehr zu sehen ist, müssen die Batterien gewechselt werden.

BATTERIETYPE: Ihr Rechner arbeitet mit vier Quecksilberbatterien Typ MR44 (1,35V) oder gleichwertigen.

BATTERIEWECHSEL:

(1) Schalten Sie den Rechner aus.

(2) Entfernen Sie die vier Schrauben vom Rückdeckel mit Hilfe eines kleinen Schraubendrehers (Abb. 1).

(3) Sorgen Sie durch Berühren eines geerdeten Körpers, etwa einer Wasserleitung, für den Abfluß eventueller statischer Aufladungen. Die Hände sollten trocken und möglichst schweißfrei sein.

(4) Heben Sie den Rückdeckel an der Schraubenseite etwas an. Er läßt sich nun leicht nach vorne wegschieben.

(5) Reinigen Sie die Oberfläche der neuen Quecksilberbatterien mit einem trockenen Tuch.

(6) Tauschen Sie *alle vier* Batterien gegen neue aus (Abb. 2).

(7) Stecken Sie die Klauen des Rückdeckels in die Schlitze am Rechner. Anschließend drücken Sie den Rückdeckel leicht zu und befestigen ihn mit den vier Schrauben. (Abb. 3)
Ziehen Sie diese Schrauben dabei fest an.

(8) Durch die Unterbrechung der Stromversorgung werden Daten, Programme und Steuerbefehle gelöscht. Um den Rechner wieder betriebsbereit zu machen, drückt man im Anschluß an den Batteriewechsel mit einer Kugelschreiberspitze den Schalter mit der Aufschrift "ALL RESET" in der Rückwand (Abb. 4). Bereits geringer Druck genügt. Bitte verwenden Sie keinen Bleistift oder andere Gegenstände, deren Spitzen abbrechen können. Der Rechner ist nun betriebsbereit.

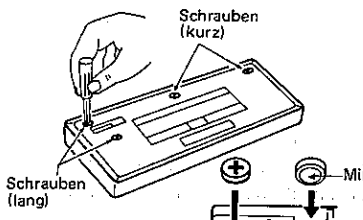


Abb. 1

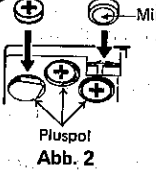


Abb. 2

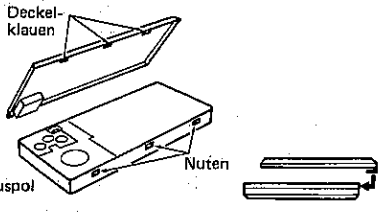


Abb. 3

HINWEIS:

Bei einem Batteriewechsel beachten Sie unbeingt folgende Hinweise:

- Wechseln Sie alle 4 Batterien gleichzeitig aus.
- Ersetzen Sie nur ungebrauchte Batterien.
- Benutzen Sie keine unterschiedlichen Batterie typen gleichzeitig.
- Der Computer arbeitet nicht mit der Silberoxid oder Alkalimanganbatterie. Diese Batterien sind ähnlich der Quecksilberbatterie. Verwechseln Sie diese nicht miteinander.

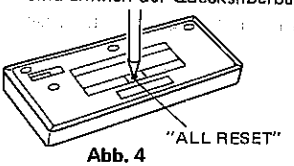
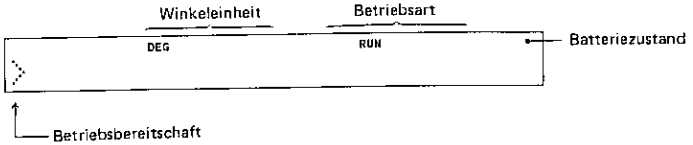


Abb. 4

4. EINSCHALTEN DES RECHNERS

Die Einschalttaste [ON] links oben im Bedienungsfeld wird kurz gedrückt. In der Anzeige erscheinen folgende Symbole:

(1) Das Symbol ⟩ zeigt an, daß der Rechner für die Aufnahme neuer Informationen bereit ist.
(2) DEG, RAD oder GRAD gibt die Einheit an, in der Winkel vom Rechner verarbeitet werden.
(3) DEF, RUN, PRO oder RESERVE zeigen die gewählte Betriebsart an.
(4) Ein Punkt informiert den Benutzer, daß die Batterie eine ausreichend hohe Spannung liefert.



Dank Permanentspeicher zeigt der Rechner jeweils die gleichen Symbole, die vor dem manuellen oder automatischen Ausschalten in der Anzeige standen. Nach einem Batteriewechsel wählt der Computer die Winkeleinheit DEG und die Betriebsart RUN.
Etwa 7 Minuten nach der letzten Bedienung einer Taste schaltet der Rechner automatisch aus. Trotzdem sollte man zur Erhöhung der Lebensdauer der Batterien den Rechner sofort nach Gebrauch mit der Taste [OFF] links oben in der Anzeige ausschalten. Alle gespeicherten Programme bleiben auch nach dem Ausschalten erhalten.

5. WAHL DER BETRIEBSART

Die Wahl der Betriebsart erfolgt mit der Taste [MODE] rechts oben im Bedienungsfeld. Jeder Druck auf [MODE] schaltet auf die nächste Betriebsart um, von DEF über RUN und PRO auf RESERVE und wieder zurück auf DEF u.s.w.
Folgende Betriebsarten sind vorgesehen:

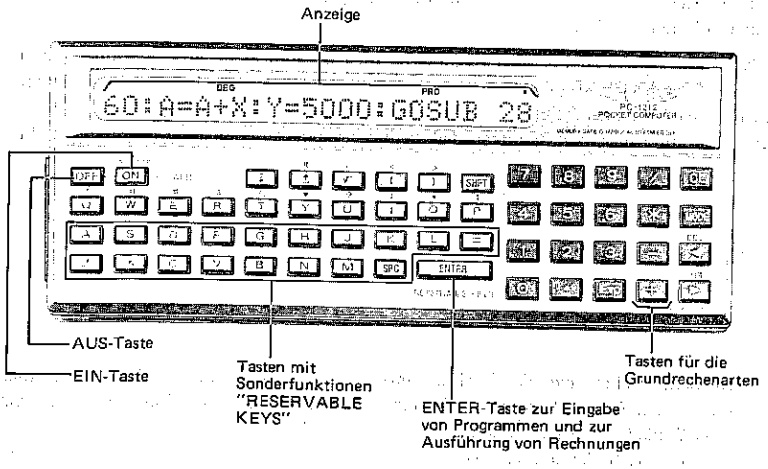
(1) **DEF** — **Definable mode**
In der Betriebsart DEF werden die mit den Namen "A" bis "SPC" versehenen Programme aufgerufen und ausgeführt. Die erlaubten Namen sind am Computer als "RESERVABLE KEYS" bezeichnet.

(2) **RUN** — **Run mode**
Berechnung von mathematischen Ausdrücken und Auswertung der Programme.

(3) **PRO** — **Program mode**
Die Betriebsart PRO dient der Eingabe von BASIC-Programmen.

(4) **RESERVE** — **Reserve program mode**
In der Betriebsart RESERVE können den am Rechner als "RESERVABLE KEYS" bezeichneten Tasten Befehle oder kleine Programme zugeordnet werden.

6. DIE BEDIENUNGSELEMENTE

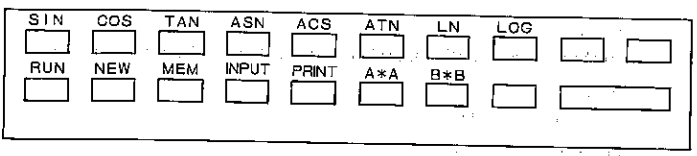


Das Bedienungsfeld besteht aus 57 Druckpunkttasten, die übersichtlich in drei Blöcke gegliedert sind.

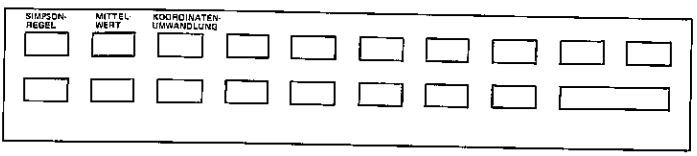
Bei einfachem Tastendruck gelten die *auf* den Tasten stehenden Bezeichnungen. Die *über* den Tasten stehenden Funktionen ruft man durch vorherigen Druck auf die Doppelfunktionstaste SHFT ab. Wurde SHFT gedrückt, so erscheint zur Kontrolle in der Anzeige der Schritzung SHFT . Nochmaliges Bedienen der Taste SHFT löscht den Doppelfunktionsbefehl. Im Text wird die tatsächlich abgerufene Funktion gedruckt. Bedient man etwa die Tasten SHFT und π , so wird die Funktion π abgerufen und man schreibt daher SHFT π .

Zur Grundausstattung des Computers gehören zwei Abdeckschablonen. Auf ihnen können die den "RESERVABLE KEYS" zugeordneten Befehle (Beispiel 1) oder Programme (Beispiel 2) notiert werden. Nähere Einzelheiten über diese Zuordnung finden Sie in den Kapiteln V9 und VI.

Beispiel 1:



Beispiel 2:



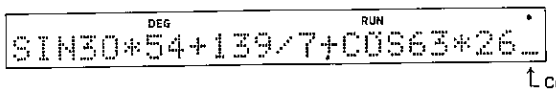
II DIE ANZEIGE

1. DIE PUNKTMATRIXANZEIGE

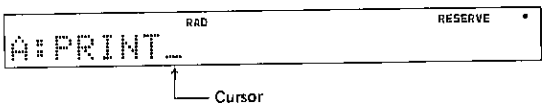
Der PC-1212 besitzt eine 24-stellige alphanumerische Flüssigkristall-Anzeige. Die Buchstaben und Zahlen werden in einer 5 x 7-Punktmatrix dargestellt. Nach jedem Tastendruck erscheint das zugehörige Symbol. Geschrieben wird von links nach rechts. Programme und mathematische Ausdrücke können also im Klartext gelesen, überprüft und korrigiert werden. Eine Marke, der Cursor (Läufer), zeigt die Stelle, auf welche die nächste Information geschrieben wird. Der Cursor steht normalerweise auf der ersten freien Stelle rechts neben dem Ausdruck und wird durch einen kurzen waagrechten Strich dargestellt. Wird der Cursor jedoch mit Hilfe der Taste [◄] oder Rückruf der Anzeige auf ein Zeichen gerückt, so erkennt man seine Position am Blinken dieser Stelle. Es erscheinen abwechselnd das markierte Zeichen und alle 35 Punkte der Matrix. Da das Eingaberegister 80 Zeichen speichern kann, die Anzeige jedoch nur 24 Stellen besitzt, verwendet der Rechner für längere Ausdrücke das Rollschreibverfahren: Neu eingegebene Zeichen werden rechts angefügt und die übrige Anzeige nach links verschoben, sodaß die linken Zeichen verschwinden. Möchte man den nicht sichtbaren Teil der Eingabe in die Anzeige zurückholen, bedient man die Taste [◄] . Ein kurzer Druck verschiebt die Anzeige um einen Schritt; hält man die Taste fest, so wird die Anzeige schnell nach rechts gerollt. Versucht man mehr als 80 Schritte einzulesen, werden diese vom Rechner nicht mehr angenommen. Jedes zusätzliche Symbol überschreibt das vorhergehende. Als Warnsignal blinkt der Cursor auf dem letzten Zeichen.
Die untenstehenden Abbildungen zeigen einige Beispiele für die Darstellung mathematischer Ausdrücke und von Programmen.

(1) Anzeige beim Einlesen eines Ausdrucks

(1.1) Betriebsart RUN

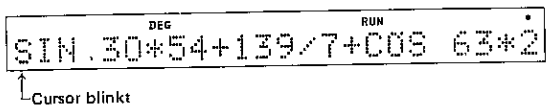


(1.2) Betriebsart RESERVE

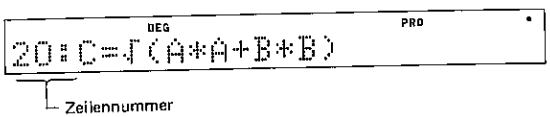


(2) Anzeige nach dem Rückruf einer Information

(2.1) Betriebsart RUN



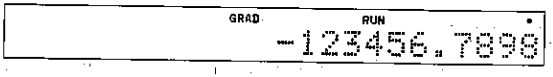
(2.2) Betriebsart PRO
Die Zahl vor dem Programm ist die Zeilennummer



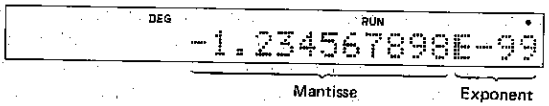
(3) Anzeige der Rechenergebnisse

Zahlenwerte werden rechtsbündig geschrieben.

(3.1) Festkommaformat

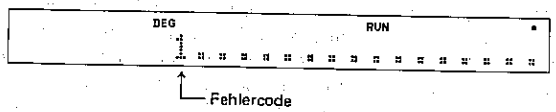


(3.2) Wissenschaftliches Format (Potenzschreibweise)



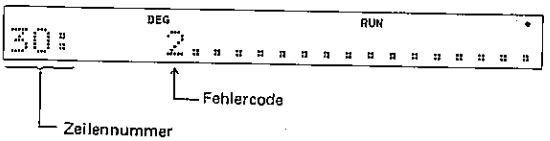
(4) Fehleranzeige

(4.1) Rechnen ohne Programmverwendung



(4.2) Programmfehler

Die Zeilennummer gibt an, in welcher Programmzeile der Rechner einen Fehler entdeckt hat.



2. SYMBOLE IN DER ANZEIGE

Über der Matrix-Schrift erscheinen je nach Betriebszustand Symbole, die zusätzliche Informationen über Eingabe und Rechenablauf bieten.

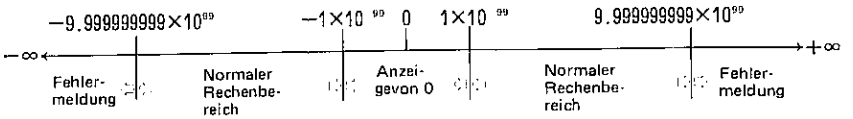
SHFT	Das Symbol SHFT erscheint, nachdem die Doppelfunktionstaste gedrückt wurde.
DEG	Winkel werden in der Einheit Grad verarbeitet.
RAD	Winkel werden in der Einheit Radiant verarbeitet.
GRAD	Winkel werden in der Einheit Gon verarbeitet.
DEF	Betriebsart DEF.
RUN	Betriebsart RUN.
PRO	Betriebsart PRO.
RESERVE	Betriebsart RESERVE.
•	Batteriezustandsanzeige. Wenn der Punkt nicht mehr zu sehen ist, sollte man die Batterien austauschen.

Um den Benutzer beim Eintippen der Beispiele auf den jeweils einzustellenden Status des Rechners hinzuweisen, werden die in der Anzeige stehenden Symbole in der ersten Zeile der Spalte "Eingabe" aufgeführt. "RUN" "DEG" bedeuten also "Betriebsart RUN, Winkeleinheit DEG (Grad)".

III ZAHLENEINGABE

1. ZAHLENBEREICH

Der Rechner arbeitet mit Zahlen, deren Betrag von 0 bis 10^{100} reicht. Ist der Betrag des Ergebnisses einer Rechnung größer $9{,}999999999 \times 10^{99}$, so meldet der Rechner diese Bereichsüberschreitung mit dem Fehlercode 1. Zahlen, deren Betrag kleiner 10^{-99} ist, werden als Null behandelt.



2. EINGABE

Eine Zahl mit maximal zehn Ziffern wird einschließlich des Dezimalpunktes so eingetastet, wie man sie schreibt. Tastet man mehr als zehn Ziffern ein, so werden die überzähligen zwar angezeigt, beim Rechnen wird jedoch nur ihr Stellenwert beachtet. Der Rechner wählt also immer die richtige Zehnerpotenz. Eine Null vor dem Komma darf weggelassen werden. Wünscht man einen negativen Zahlenwert, so drückt man *vor* der Eingabe von Ziffern die Taste [−] . Das Vorzeichen erscheint links von der Zahl.
Möchte man Zahlen im wissenschaftlichen Format $A \times 10^B$ einlesen, drückt man nach dem Eintasten der maximal zehnstelligen Mantisse A die Taste [EXP] und gibt dann den maximal zweistelligen Exponenten B mit Vorzeichen ein. Tastet man mehr als zwei Exponentenziffern ein, werden beim Rechnen lediglich die beiden letzten beachtet.

Beispiele:

Zahl	Eingabe
−12.345	[−] [1] [2] [•] [3] [4] [5]
6.7×10^8	[6] [•] [7] [Exp] [8]
-9.12×10^{-34}	[−] [9] [•] [1] [2] [Exp] [−] [3] [4]

Eingetastete Zahl	Vom Rechner angenommene Zahl
1234567898765	$1.234567898 \times 10^{12}$
9.87654321234	9.876543212
0.0000000002345678	2.345678×10^{-10}
0.00001234567 [Exp] 24	1.234567×10^{19}
3 [Exp] 123	3×10^{23}
4 [Exp] [−] 3210	4×10^{-10}

3. ANZEIGEFORMAT

Zahlen können im Festkommaformat oder im wissenschaftlichen Format angezeigt werden. Im Programmbetrieb wird das Format einschließlich der Anzahl der Nachkommastellen mit der Formatanweisung USING gewählt. Beim nichtprogrammierten Rechnen hängt das Format von Größe und Stellenanzahl des anzuzeigenden Wertes ab.

(1) Unabhängig von der Anzeige werden Zahlen in den rechnerinternen Registern immer in der Form $A \times 10^B$ gespeichert wobei $1 \leq |A| \leq 9{,}999999999$ und $-99 \leq B \leq 99$ ist.

(2) Zahlen, deren Betrag größer 999999999 oder kleiner 0,000000001 ist, werden immer im wissenschaftlichen Format angezeigt.

(3) Zahlen mit einem Betrag zwischen 10^{-9} und 999999999 werden im Festkommaformat angezeigt, wenn das innerhalb von zehn Stellen ohne Beschneidung der Nachkommastellen möglich ist. Würden dagegen Nachkommastellen unterdrückt, wählt der Rechner das wissenschaftliche Format.

Beispiel:

Rechnerinternes Ergebnis	Angezeigte Zahl
1.234×10^{-4}	0.0001234
$1.23456789 \times 10^{-4}$	1.23456789 E-4

IV. RECHNEN OHNE PROGRAMMUNTERSTÜTZUNG

1. MANUELLES UND PROGRAMMIERTES RECHNEN

Die Leistung des Computers PC-1212 wird erst im Programmbetrieb voll ausgeschöpft. Normalerweise wird daher in der Betriebsart PRO programmiert und das Programm in der Betriebsart RUN oder DEF abgearbeitet.
Einfache Probleme jedoch, deren Programmierung nicht lohnt, oder Rechnungen, die nur einmal auszuführen sind, können in den Betriebsarten RUN, DEF oder RESERVE direkt gelöst werden. Der Computer kann also auch als normaler wissenschaftlicher Taschenrechner verwendet werden.

> Bei den folgenden Rechenbeispielen wird die Betriebsart RUN verwendet. Stellen Sie daher den Rechner mit Hilfe der Taste [MODE] auf die Betriebsart RUN. In der Anzeige muß das Symbol RUN stehen.

Die Eingabe hat beim manuellen Rechnen folgende allgemeine Form:

Mathematischer Ausdruck [ENTER]

Beispiel: 5 [*] 4 [ENTER]

Nach Eingabe der Formel verwendet man also die Eingabetaste [ENTER] und nicht etwa [=] . Die Taste [=] dient nur der Zuordnung von Zahlen und Anweisungen zu bestimmten Speichern.

Für die Formulierung des mathematischen Ausdrucks stehen folgende Elemente zur Verfügung:

Zifferntasten	0 bis 9, • , π , Exp
Vorzeichen	+, – (Das positive Zeichen wird üblicherweise weggelassen)
Rechenoperationen	+, –, * (Multiplikation), / (Division), ∧ (allgemeine Exponentialfunktion)

Vergleichsoperatoren	=, >, <, >=, <=, <>
Mathematische Funktionen	SIN, COS, TAN, ASN, ACS, ATN, LN, LOG, EXP, DEG, DMS, INT, ABS, SGN, $\sqrt{}$
Klammern	(,)
Speicher	A bis Z, A (nn)

Benötigt man häufig spezielle Funktionen, etwa trigonometrische Funktionen, Hyperbelfunktionen oder den Logarithmus zur Basis 2, so kann man diese Funktionen den "RESERVABLE KEYS" zuordnen und ebenfalls auf Tastendruck abrufen. Nähere Einzelheiten sind im Kapitel VI beschrieben.

Im Folgenden wird allgemein von einem mathematischen Ausdruck gesprochen, auch dann, wenn es sich im speziellen Fall nur um Zahlen oder Speicherinhalte handlet, also etwa π oder A.

Obwohl es nicht in jedem Fall notwendig ist, empfiehlt es sich, vor jeder neuen Rechnung mit Hilfe der Taste [CL] die Anzeige zu löschen.

Die meisten Befehle können zur Vereinfachung der Eingabe in abgekürzter Form geschrieben werden. So kann man etwa zur Wahl der Winkeleinheit Grad DEG. anstelle von DEGREE schreiben. In der Anleitung werden jedoch immer die ungekürzten Befehle verwendet.

2. GRUNDRECHENARTEN

Da der Rechner entsprechend den Rechenregeln der Algebra die Grundrechenarten Multiplikation und Division vorrangig vor Addition und Subtraktion ausführt, kann man alle Rechnungen so, wie sie im Ansatz stehen, eintasten. Es müssen also keine zusätzlichen Klammern verwendet werden.

In der Computersprache BASIC bedeuten: * : Multiplikation
/ : Division
E : Zehnerexponent

2.1 Addition und Subtraktion

Beispiel: 7 − 9 + 14 =
−4.2 + 5 − 12.3 =

Eingabe	Anzeige	Anmerkung
"RUN"	>	
[CL] 7 [−] 9 [+] 14	7−9+14_	Math. Ausdruck
[ENTER]	12.	Ergebnis
[CL] [−] 4.2 [+] 5 [−] 12.3	−4. 2+5−12. 3_	Math. Ausdruck
[ENTER]	−11. 5	Ergebnis

2.2 Multiplikation und Division

Beispiel: 12*24/5= (12×24÷5=)
27E3*4/12= (27×10³×4÷12=)

Eingabe	Anzeige	Anmerkung
[CL] 12 [*] 24 [/] 5	12*24/5_	Math. Ausdruck
[ENTER]	57. 6	Ergebnis
[CL] 27 [Exp] 3 [*] 4 [/] 12	27E3*4/12_	Math. Ausdruck
[ENTER]	9000.	Ergebnis

2.3 Gemischte Rechnungen

Beispiel: 54+24.3*16.49/3.4−37.4=

Eingabe	Anzeige	Anmerkung
CL 54 + 24.3 * 16.49	54+24.3*16.49_	
/ 3.4 − 37.4	54+24.3*16.49/3.4−37.4_	
ENTER	134.455	

2.4 Verwendung eines Ergebnisses in der nachfolgenden Rechnung

Wird vor einer neuen Rechnung nicht mit CL gelöscht, kann man die in der Anzeige stehende Zahl für die Fortführung der Rechnung mit den Befehlen +, −, *, / oder ^ verwenden.

Beispiel: ① 3 + 4 = ② 7 − 5 + 6 =

Das Ergebnis der Rechnung ①, also 7 wird in der Rechnung ② verwendet, sodaß die Rechnung 3 + 4 − 5 + 6 ausgeführt wird.

Eingabe	Anzeige	Anmerkung
CL 3 + 4 ENTER	7.	Ergebnis des Ausdrucks ①.
−	7. −_	Das Ergebnis von ①, 7 wird als erste Zahl von ② verwendet.
5 + 6	7. −5+6_	
ENTER	8.	

3. DIE ALLGEMEINE EXPONENTIALFUNKTION

Die allgemeine Exponentialfunktion wird entsprechend den Regeln der Mathematik vorrangig vor den Grundrechenarten ausgeführt.

Beispiele: 4^3= (4^3=)

3^3.2*4^2.4= ($3^{3.2}\times 4^{2.4}$=)

4^3^2= (4^{3^2}=)

Eingabe	Anzeige	Anmerkung
CL 4 SHIFT ^ 3	4^3_	Ausdruck
ENTER	64.	Ergebnis
3 SHIFT ^ 3.2 *	3^3.2*_	Ausdruck
4 SHIFT ^ 2.4	3^3.2*4^2.4_	
ENTER	936.9836103	Ergebnis
4 SHIFT ^ 3 SHIFT ^ 2	4^3^2_	Ausdruck
ENTER	262144.	Ergebnis

Bei der allgemeinen Exponentialfunktion dürfen nur 3 Ebenen verwendet werden. Ein Ausdruck A ∧ B ∧ C ∧ D ist nicht zulässig.

Hinweis
Bei konstanten mit negativer Basis wird die Exponentialfunktion nicht vorzeichenrichtig ausgeführt. So wird z. B. -2 ∧ 2 vom PC-1212 als-(2 ∧ 2) aufgefaßt.
Bei variablen mit negativer Basis wird die Exponentialfunktion nicht ausgeführt. Bei z. B. A ∧ B (A = -2, B = 2) wird ein Fehler gemeldet.
Durch Verwendung des Absolutwertes von (A), z. B. (ABS A) ∧ B wird die Exponentialfunktion ausgeführt, das Vorzeichen wird jedoch nicht berücksichtigt.

4. KLAMMERRECHNUNG

Ihr Rechner hat 15 Klammerregister. Die Klammern können beliebig geschachtelt werden; sie dürfen im Exponenten stehen und Funktionswerte enthalten. Öffnet man mehr als 15 Klammern, erscheint nach Abruf des Ergebnisses in der Anzeige die Fehlermeldung. Die Taste [ENTER] schließt automatisch alle vorher geöffneten Klammern; in der Anzeige steht sofort das Endergebnis.

Da der Rechner alle algebraischen Regeln beherrscht, dürfen die Klammern genau so gesetzt werden, wie sie im Ansatz stehen.

In der üblichen Schreibweise läßt man das Malzeichen vor Klammern häufig weg. Bei einem Rechner darf das nicht praktiziert werden, da er sonst nicht weiß, welche Rechenoperation mit dem Klammerausdruck ausgeführt werden soll.

Besondere Vorsicht ist beim Gebrauch von Klammern in Verbindung mit Brüchen und Wurzeln angebracht, wie folgende Beispiele zeigen:

$A+B/C \rightarrow A+\frac{B}{C}$ $\qquad \sqrt{\ }A+B \rightarrow \sqrt{A}+B$

$(A+B)/C \rightarrow \frac{A+B}{C}$ $\qquad \sqrt{\ }(A+B) \rightarrow \sqrt{A+B}$

$A/C*D \rightarrow \frac{AD}{C}$ $\qquad \sqrt{\ }A*B \rightarrow B\sqrt{A}$

$A/(C*D) \rightarrow \frac{A}{CD}$ $\qquad \sqrt{\ }(A*B) \rightarrow \sqrt{AB}$

$A/B/C \rightarrow \frac{\frac{A}{B}}{C}=\frac{A}{BC}$ $\qquad A*B+C \rightarrow AB+C$

$A/(B/C) \rightarrow \frac{A}{\frac{B}{C}}=\frac{AC}{B}$ $\qquad A*(B+C) \rightarrow A(B+C)$

Beispiel: $(72+9)/4*\{21*(68/(7-3)+2)\}=$

Eingabe	Anzeige
[CL] [(]72[+]9[)][/]4[*]	(72+9) /4*_
[(]21[*][(]68[/][(]7	(72+9) /4* (21* (68/ (7_
[-]3[)][+]2[)][)]	+9) /4* (21* (68/ (7-3) +2)) _
[ENTER]	8079. 75

5. DIE MATHEMATISCHEN FUNKTIONEN

Auch beim Rechnen mit mathematischen Funktionen folgt die Dateneingabe exakt der entsprechend den Regeln der Algebra geschriebenen Formel. Irgendwelche Umstellungen sind nicht nötig. Diese optimale Eingabemethode kann bei Benutzern, die konventionelle Rechner gewohnt sind, zu Anfangsschwierigkeiten führen. Der PC-1212 verwendet nämlich nicht einmal bei der Funktionsberechnung die "Polnische Notation". Der Wert $\sqrt{10}$ oder ln 20 wird wie im Ansatz [√] 10 oder [L] [N] 20 geschrieben und nicht wie sonst üblich 10 [√] oder 20 [LN] . Bei der Ermittlung der Funktionswerte von Ausdrücken verwendet man Klammern z.B. SIN (π/2).

Zur Berechnung der Funktionswerte besizt der Rechner eigene Rechenwerke. Alle gespeicherten Daten bleiben daher erhalten. Das hat den Vorteil, daß man Grundrechenarten und Klammerrechnungen unter Einbeziehung von Funktionswerten ausführen kann.

Die Berechnung der Funktionswerte nimmt bis zu einer Sekunde in Anspruch. Während dieser Zeit zeigt die Anzeige allein "RUN" und es werden keine Daten angenommen.

5.1 Quadratwurzel ($\sqrt{\ }$)

Beispiel: $\sqrt{73}=$ ($\sqrt{73}=$)

$\sqrt{\sqrt{256}}=$ ($\sqrt{\sqrt{256}}=\sqrt[4]{256}=$)

$\sqrt{(3*3+4*4)}=$ ($\sqrt{3^2+4^2}=$)

Eingabe	Anzeige	Anmerkung
[CL] [√] 73	√73_	
[ENTER]	8.544003745	
[√] [√] 256	√√256_	
[ENTER]	4.	
[√] [(] 3 [*]	√(3*_	
3 [+] 4 [*] 4 [)]	√(3*3+4*4)_	
[ENTER]	5.	

5.2 Exponentialfunktion (EXP)

Die Exponentialfunktion $e^x = \exp x$ wird am Rechner mit der Tastenfolge [E] [X] [P] abgerufen.

Beispiel: $e^{-13.6} = \exp -13.6 =$

Eingabe	Anzeige	Anmerkung
[CL] [E] [X] [P] [−] 13.6	EXP −13.6_	
[ENTER]	1.24049508E−06	

5.3 Logarithmusfunktionen (LN, LOG)

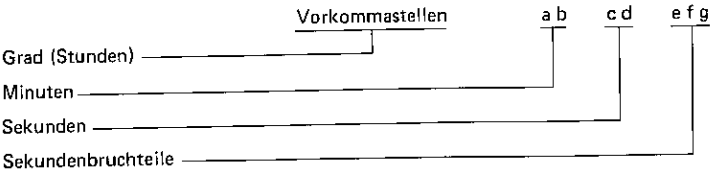
Natürlicher Logarithmus ln und Zehnerlogarithmus (gewöhnlicher oder Briggsscher Logarithmus) lg werden mit der Tastenfolge [L] [N] und [L] [O] [G] abgerufen.

Beispiele: ln 7.4 =
lg 100 =

Eingabe	Anzeige	Anmerkung
[CL] [L] [N] 7.4	LN7. 4_	
[ENTER]	2. 00148	
[L] [O] [G] 100	LOG100_	
[ENTER]	2.	

5.4 Umrechnung Dezimalsystem ↔ Sexagesimalsystem (DMS, DEG)

Bei diesen Umrechnungen werden sexagesimal geteilte Winkel oder Zeiten in folgendem Format verarbeitet:



Eingabeablauf bei der Umrechnung Dezimalsystem ⇒ Sexagesimalsystem:

[D] [M] [S] Dezimalzahl [ENTER]

In der Anzeige steht die Sexagesimalzahl

Eingabeablauf bei der Umrechnung Sexagesimalsystem ⇒ Dezimalsystem:

[D] [E] [G] Sexagesimalzahl [ENTER]

In der Anzeige steht die Dezimalzahl

Beispiele: Umrechnung von 15.4125° in Grad/Minuten/Sekunden
Umrechnung von 15°24'45" in Dezimalgrad

Eingabe	Anzeige	Anmerkung
[CL] [D] [M] [S] 15.4125	DMS15. 4125_	
[ENTER]	15. 2445	15°24' 45"
[D] [E] [G] 15.2445	DEG15. 2445_	
[ENTER]	15. 4125	15.4125°

5.5 Trigonometrische Funktionen (SIN, COS, TAN)

Die trigonometrischen Funktionen können von Winkeln in allen üblichen Einheiten berechnet werden. Die Einheiten wählt man wie folgt:

Winkeleinheit	Symbol	Eingabemodus
Grad (Altgrad, °)	DEG	[D] [E] [G] [R] [E] [E] [ENTER]
Radiant (Bogenmaß, rad)	RAD	[R] [A] [D] [I] [A] [N] [ENTER]
Gon (Neugrad, gon)	GRAD	[G] [R] [A] [D] [ENTER]

Ist ein Winkel sexagesimal geteilt, also in Grad, Minuten und Sekunden gegeben, so muß man ihn vor Anwendung der Winkelfunktionstasten in einen dezimal geteilten Winkel umrechnen. Um Verwechslungen vorzubeugen, beachte man, daß die bei der Rechnerbeschriftung verwendete englische Abkürzung GRAD für gradian (Neugrad) mit der deutschen Normbezeichnung Grad für Altgrad übereinstimmt.

Beispiele: sin 30° = Winkeleinheit DEG
cos (π/4) = Winkeleinheit RAD
tan 150 gon = Winkeleinheit GRAD

Eingabe	Anzeige	Anmerkung
"DEG"		
[CL] [S] [I] [N] 30 [ENTER]	SIN30_	Wahl der Einheit DEG
	0. 5	sin 30°
"RAD"		
[C] [O] [S] [(] [SHIFT] [π]	COS (π _	
[/] 4 [)]	COS (π/4) _	
[ENTER]	7. 071067812E−01	cos (π/4) rad
"GRAD"		
[T] [A] [N] 150	TAN150_	
[ENTER]	−1.	tan 150 gon

5.6 Arcusfunktionen (ASN, ACS, ATN)

Die Arcusfunktionen arcsin, arccos und arctan werden mit der Tastenfolge [A] [S] [N], [A] [C] [S], und [A] [T] [N] abgerufen. Die Wahl der Winkeleinheit erfolgt wie bei den trigonometrischen Funktionen.

Beispiele: arcsin (−0.5) = Winkeleinheit DEG
arccos (−0.5 + 0.1) = Winkeleinheit RAD
arctan (7/3) = Winkeleinheit GRAD

Eingabe	Anzeige	Anmerkung
"DEG"		
[CL] [A] [S] [N] [−] .5	ASN−. 5_	
[ENTER]	−30.	Grad
"RAD"		
[A] [C] [S] [(] [−] .5	ACS (−. 5_	
[+] .1 [)]	ACS (−. 5+. 1) _	
[ENTER]	1. 982313173	Radiant
"GRAD"		
[A] [T] [N] [(] 7	ATN (7_	
[/] [3] [)]	ATN (7/3) _	
[ENTER]	74. 22378832	Gon

5.7 Ganzzahliger Anteil einer Zahl, Integer (INT)

Die Funktion INT x bewirkt, daß die größte ganze Zahl, die kleiner oder gleich x ist, ermittelt wird. Es ist also INT 5.12 = 5; INT 5.99 = 5; INT (−5.12) = −6; INT (−5.99) = −6.

Beispiele: INT (65/3) =
INT (−0.3) =

Eingabe	Anzeige	Anmerkung
[I] [N] [T] [(] 65 [/] 3 [)]	INT (65/3) _	
[ENTER]	21.	
[I] [N] [T] [−] .3	INT −. 3 _	
[ENTER]	−1.	

5.8 Signum (SGN)

Die Signumfunktion SGN x gibt das Vorzeichen des Arguments x in folgender Weise an:

sgn x = +1 für $x > 0$
sgn x = 0 für $x = 0$
sgn x = −1 für $x < 0$

Beispiel: sgn (5 − 9) =

Eingabe	Anzeige	Anmerkung
[S] [G] [N] [(] 5 [−] 9 [)]	SGN (5−9) _	
[ENTER]	−1.	

5.9 Absolutbetrag (ABS)

Die Funktion ABS x ermittelt den Absolutbetrag $|x|$ des Arguments x.

Beispiel: $|5-9| =$

Eingabe	Anzeige	Anmerkung
A B S (5 − 9) ENTER	ABS (5−9) _ 4.	

6. VERGLEICHSOPERATOREN

BASIC kennt sechs Vergleichsoperatoren: $x<y$; $x\leqq y$; $x=y$; $x\geqq y$; $x>y$ und $x\neq y$. Allgemein schreibt man für einen beliebigen der sechs Operatoren das Zeichen $\circ$; also $x \circ y$. Der Rechner meldet mit der Antwort 1, daß die Zahlen x und y die in der Vergleichsoperation angegebene Bedingung erfüllen. Wird die Bedingung nicht erfüllt, ist die Antwort 0.
Probleme gibt es lediglich bei Verwendung des Zeichens = zusammen mit Speichern, da mit = auch die Zuordnung einer Zahl zu einem Speicher realisiert wird.

Die Schreibweise

Speichername = mathematischer Ausdruck ENTER

z.B. A = (2 + L N 5) ENTER

ordnet den mathematischen Ausdruck dem gewählten Speicher zu. Mit der allgemeinen Form

mathematischer Ausdruck = Speichername ENTER

fragt man dagegen ab, ob der mathematische Ausdruck mit der gespeicherten Zahl übereinstimmt. Diese Einschränkung gilt jedoch nicht beim logischen Vergleich nach einer IF-Anweisung.

Mathematisches Symbol	BASIC Schreibweise	Sprechweise	Meldung des Rechners
$<$	<	kleiner als	1, wenn $x<y$ 0, wenn $x\geqq y$
$\leqq$	<=	kleiner gleich	1, wenn $x\leqq y$ 0, wenn $x>y$
$=$	=	gleich	1, wenn $x=y$ 0, wenn $x\neq y$
$\geqq$	>=	größer gleich	1, wenn $x\geqq y$ 0, wenn $x<y$
$>$	>	größer	1, wenn $x>y$ 0, wenn $x\leqq y$
$\neq$	<>	ungleich	1, wenn $x\neq y$ 0, wenn $x=y$

Beispiele: $(5 + 8) > (3 * 4)$?
$(24/5) \leq (2.4 * 2)$?

Eingabe	Anzeige	Anmerkung
(5 + 8) SHIFT >	(5+8) > _	
(3 * 4)	(5+8) > (3*4) _	
ENTER	1.	Wahr
(24 / 5) SHIFT < =	(24/5) <=_	
(2.4 * 2)	(24/5) <= (2. 4*2) _	
ENTER	1.	Wahr

Auch die logischen Verknüpfungen ODER (OR) und UND (AND) lassen sich ausführen.

Die ODER-Verknüpfung erreicht man mit der allgemeinen Form

Logischer Vergleich + Logischer Vergleich ENTER

Das Beispiel $(A < 0) + (A > 8)$ etwa liefert die Kennzahl 1, "wahr", wenn A kleiner 0 *oder* größer 8 ist; die Kennzahl 0, "falsch" dagegen, wenn A zwischen 0 und 8 liegt.

Die UND-Verknüpfung erreicht man mit der allgemeinen Form

Logischer Vergleich * Logischer Vergleich ENTER

Das Beispiel $(B > 1) * (C < 6)$ etwa liefert die Kennzahl 1, wenn B größer 1 *und* C kleiner 6 sind; die Kennzahl 0 dagegen, wenn B kleiner 1 oder C größer 6 werden.

7. KONSTANTENSPEICHER

Der Rechner besitzt 26 permanente Konstantenspeicher mit den Namen A bis Z oder indizierte Konstantenspeicher mit den Namen A(1) bis A(26). Dabei gelten folgende Identitäten: $A \equiv A(1)$; $B \equiv A(2)$; · · · · · $Z \equiv A(26)$. Darüberhinaus kann man weitere Speicher auf Kosten von Programmschritten mit den Namen A(27) bis A(204) beim PC-1212 definieren. Jeder Speicher belegt 8 Programmschritte.

7.1 Speichern von Zahlen und Rechenergebnissen

Mathematische Ausdrücke werden in der folgenden allgemeinen Form in die Speicher eingelesen:

Speichername = Mathematischer Ausdruck ENTER

Dabei wird die ursprünglich im Speicher stehende Zahl überschrieben; der Speicher muß also nicht gelöscht werden.

Beispiele:

A = 123.4 ENTER	Die Zahl 123.4 wird in Speicher A eingelesen.
Y = 5 * 6 ENTER	Das Ergebnis der Rechnung 5 x 6, also 30, wird dem Speicher Y zugeordnet.
Z = A + B ENTER	Die Summe der in A und B gespeicherten Zahlen wird in Speicher Z gestellt.

A (26) = 3 + 9 ENTER	Das Ergebnis der Rechnung 3 + 9, also 12 wird in A(26) gespeichert.
A = A + 1	Der Inhalt von A wird um 1 erhöht.

7.2 Abruf der gespeicherten Zahlen

Die allgemeine Form des Speicherabrufs lautet:

Speichername ENTER

Beispiele:

A ENTER	Der Inhalt von Speicher A wird abgerufen.
A (18) ENTER	Der Inhalt von Speicher A(18), der mit Speicher R identisch ist, wird abgerufen.

7.3 Rechnen mit Speichern

Die Speicherinhalte kann man auch durch Eingabe der Speichernamen ohne nachfolgende Bedienung der ENTER Taste anwählen. Dann stehen in der Anzeige die Namen der Speicher. Bei Berechnungen werden aber die den Namen zugeordneten Zahlenwerte in den Speichern verwendet. Steht zwischen zwei Speichernamen oder zwischen Zahlenwert und Namen kein Rechenbefehl, so interpretiert das der Rechner wie in der Mathematik üblich als Multiplikationsbefehl. Will man etwa den Zahlenwert im Speicher A mit dem Inhalt des Speichers B multiplizieren, so erreicht man das mit der Folge A B ENTER . Selbstverständlich führt auch die etwas längere Eingabe A * B ENTER zum richtigen Ergebnis. Analog darf man auch 2A, 3π, BA(12), SIN 2A oder 2π A$\wedge$3 anstelle von 2 * A, 3 * π, B * A(12), SIN (2 * A) oder 2 * π * (A $\wedge$ 3) schreiben. Bei indizierten Konstanten darf der Index ein mathematischer Ausdruck sein; A (2 + 3) zum Beispiel kennzeichnet den Speicher A(5), der mit Speicher E identisch ist.

Beispiele:

Eingabe	Anzeige
A = 4 ENTER	4
B = 5 ENTER	5
A + B * B − 12 ENTER	17
√ (A + B) ENTER	3

7.4 Löschen der Konstantenspeicher

Die Anweisung C L E A R ENTER löscht sämtliche Konstantenspeicher. Einzelne Speicher löscht man mit der allgemeinen Form

Speichername = 0 ENTER

8. TEXTSPEICHER (STRINGS)

Zum Speichern von Texten bis zu 7 Buchstaben verwendet man die Konstantenspeichernamen mit dem Zusatz Dollar ($). Die Speichernamen lauten also A$ bis Z$ und A$(1) bis Z$(26). Auch hier kann man Speicherplätze auf Kosten von Programmschritten schaffen. Speicher können nur entweder als Konstantenspeicher oder als Textspeicher verwendet werden. Der Text muß in Anführungszeichen stehen.

8.1 Speichern von Texten

Texte werden in folgender allgemeinen Form gespeichert:

Textspeichername [=] [SHIFT] ["] Text [SHIFT] ["] [ENTER] oder
Textspeichername [=] Textspeichername [ENTER]

Beispiel: Speichern des Wortes "SHARP" in Speicher D und Kopieren in Speicher E

[D] [SHIFT] [$] [=] [SHIFT] ["] [S] [H] [A] [R] [P] [SHIFT] ["] [ENTER]
[E] [SHIFT] [$] [=] [D] [SHIFT] [S] [ENTER]

8.2 Abruf der gespeicherten Texte

Allgemeine Form:

Textspeichername [ENTER]

Beispiel:

[D] [SHIFT] [$] [ENTER] Das Wort SHARP erscheint in der Anzeige.

9. GLEICHZEITIGE EINGABE MEHRERER AUSDRÜCKE

Zwei oder mehrere Ausdrücke können gleichzeitig eingegeben werden, wenn man sie mit einem Komma trennt. Der Rechner zeigt in diesem Falle allerdings nur das Ergebnis des letzten mathematischen Ausdrucks an. Die allgemeine Form der Eingabe lautet:

Ausdruck [SHIFT] [,] Ausdruck [SHIFT] [,] ····· Ausdruck [ENTER]

Beispiel: Es sei $A = \frac{5}{12-4}$, $B = \frac{87}{24}$, $C = \frac{12}{7+8}$. Man ermittle A∗B/C

Eingabe	Anzeige	Anmerkung
[A] [=] 5 [/] [(] 12 [−] 4 [)]	A=5/(12−4)_	A
[SHIFT] [,] [B] [=] 87 [/] 24 [SHIFT] [,]	A=5/(12−4), B=87/24,_	B
[C] [=] 12 [/] [(] 7	/(12−4), B=87/24, C=12/(7_	
[+] 8 [)] [SHIFT] [,]	−4), B=87/24, C=12/(7+8),_	C
[A] [∗] [B] [/] [C]	=87/24, C=12/(7+8), A∗B/C_	
[ENTER]	2.83203125	A∗B/C

10. ZUORDNUNG VON AUSDRÜCKEN ZU TASTEN

In der Betriebsart RESERVE kann man den 18 am Tastenfeld mit RESERVABLE KEYS bezeichneten Tasten beliebige mathematische Funktionen, Ausdrücke oder Befehle zuordnen. Jeder Anwender kann so die von ihm benötigten speziellen Funktionen festlegen. Damit lassen sich zum Beispiel Hyperbelfunktionen, Koordinatentransformationen und statistische Funktionen in den Betriebsarten RUN und PRO mit einem einfachen Knopfdruck abrufen. Benötigt man etwa häufig verschiedene Winkeleinheiten, ordnet man sie den Tasten A, S und D zu und ruft dann Grad mit [SHIFT] [A], Radiant mit [SHIFT] [S] und Gon mit [SHIFT] [D] ab. Damit spart man Zeit bei der Eingabe. Notiert man weiter alle so definierten Funktionen auf der dem Rechner beiliegenden Abdeckschablone, so erhält man den optimal auf die eigenen Bedürfnisse zugeschnittenen Rechner. Nähere Einzelheiten der Funktionszuordnung und des Abrufs finden sich im Kapitel VI.

11. INFORMATIONSRÜCKRUF (PLAYBACK)

Bedient man eine der Playback-Tasten [►] oder [◄] im Anschluß an [ENTER], so wird die ursprüngliche Information wieder in die Anzeige zurückgeholt. Der Rückruf ist möglich, da der Rechner die komplette Eingabe bis zum Löschen oder zum Beginn einer neuen Berechnung im Eingaberegister speichert. Man kann so die Eingabe überprüfen und korrigieren, falls das Ergebnis unglaubhaft erscheint oder Änderungen im Ansatz gewünscht werden. Ist bei der Eingabe ein Fehler unterlaufen, den der Rechner erkennen kann, so erscheint die Fehlermeldung. Holt man den Ausdruck mit einer der Playback-Tasten zurück, so blinkt der Cursor an der Fehlerstelle und erleichtert so die Korrektur.

11.1 Rückruf ohne Fehlermeldung

Der eingetastete Ausdruck wird so angezeigt, daß sein Anfang links steht; bei langen Ausdrücken sieht man nur die ersten 24 Zeichen. Der Cursor steht auf dem ersten Zeichen des Ausdrucks. Möchte man einen Ausdruck mit mehr als 24 Zeichen überprüfen, bedient man die Taste [►] .

Beispiel:

Eingabe	Anzeige	Anmerkung
[A] [=] 19 [+] 54	A=19+54_	
[ENTER]	73.	
[►] oder [◄]	A=19+54	Rückruf

11.2 Rückruf nach Fehlermeldung

Der eingetastete Ausdruck wird so angezeigt, daß die blinkende Fehlerstelle innerhalb der Anzeige liegt. Bei langen Ausdrücken können sowohl links als auch rechts der Anzeige weitere Teile der Eingabe stehen. Diese kann man mit den Tasten [◄] und [►] in die Anzeige schieben.

Beispiel:

Eingabe	Anzeige	Anmerkung
[B] [=] [(] [(] 123456 [+]	B=((123456+_	
789012 [)] [*] 427 [/] 197	(123456+789012)*427/197_	
[)] [/] 0 [+] 139	+789012)*427/197)/0+139_	
[ENTER]	1.....	Fehler
[►] oder [◄]	3456+789012)*427/197)/0+	Rückruf

12. KORREKTUR UND ÄNDERUNG (EDITIEREN)

Alle Ausdrücke können vor der Berechnung oder nach Rückruf beliebig verändert werden. Dazu stehen vier Tasten zur Verfügung:

- [◄] Rückt den Cursor nach links.
- [►] Rückt den Cursor nach rechts.
- [INS] (INSert) schafft Platz für das Einfügen eines neuen Befehls.
- [DEL] (DELeat) entfernt überflüssige Befehle.

Lohnt eine Korrektur des Ausdrucks nicht, löscht man alles mit [CL] .

12.1 Markierung der Korrekturstelle

Mit den Tasten [►] oder [◄] schiebt man den Cursor an die Stelle, an der die Änderung erfolgen soll. Die Lage des Cursors wird durch Blinken gekennzeichnet. Ein kurzer Tastendruck verschiebt den Cursor um eine Stelle; hält man die Taste fest, legt der Cursor etwa zehn Schritte in der Sekunde zurück. Zur Überprüfung langer Ausdrücke schiebt man den Cursor im Schnellauf nahe an die gewünschte Position und bewegt ihn dann Schritt für Schritt mit den Tasten [►] oder [◄] .

12.2 Überschreiben

Nachdem man den Cursor auf das fehlerhafte Zeichen geschoben hat, drückt man die Taste mit dem richtigen Zeichen. Nach der Korrektur steht der Cursor beim folgenden Zeichen, das nun ebenfalls überschrieben werden kann.

12.3 Löschen

Das durch den Cursor markierte Zeichen wird durch Drücken der Tastenfolge [SHIFT] [DEL] gelöscht. Die gesamte Anzeige wird anschließend um eine Stelle nach links gerollt. Der Cursor behält seine Position bei, sodaß auch längere Eingaben einfach zu löschen sind. Auch durch Überschreiben der Zeichen mit [SPC] kann man löschen.

12.4 Einfügen neuer Zeichen

Man schiebt den Cursor zu dem Zeichen, *vor* das man ein neues einfügen möchte. Dann bedient man die Tastenfolge [SHIFT] [INS] . In der Anzeige erscheint vor dem angewählten Zeichen die Marke ⌴ , auf welcher der Cursor steht. Dieses Leerzeichen wird nun mit dem einzutastenden Zeichen gefüllt und verschwindet; der Cursor springt zum nächsten Zeichen. Vorsicht ist lediglich bei langen Ausdrücken angebracht. Besteht ein Ausdruck nämlich bereits aus 80 Schritten, so wird durch das Einfügen eines neuen Zeichens das 80. Zeichen gelöscht.

12.5 Abruf des Ergebnisses nach der Korrektur

Hat man einen Ausdruck korrigiert, so kann man unabhängig von der Stellung des Cursors sofort das Ergebnis mit [ENTER] abrufen.

Beispiel: Bei der Eingabe des Ausdrucks A = 5 + 6 * (21 / SIN 30) werden eine Reihe von Fehlern gemacht, die korrigiert werden sollen. Die richtige Tastenfolge lautet:

[A] [=] 5 [+] 6 [*] [(] 21 [/] [S] [I] [N] 30 [)] [ENTER]

(1) Bei der Eingabe des Ausdrucks wurde versehentlich die Taste [+] anstelle von [*] gedrückt. Der Fehler wurde sofort nach der Eingabe bemerkt.

Eingabe	Anzeige	Anmerkung
"DEG" [A] [=] 5 [+] 6 [+]	A=5+6+_	Falsche Taste.
[◄]	A=5+6+	Rücksetzen des Cursors.
[*]	A=5+6*_	Richtiger Befehl.
⋮		

(2) Das Eintasten des Befehls SIN wurde vergessen.

Eingabe	Anzeige	Anmerkung
"DEG"		
A = 5 + 6 *	A=5+6*_	
(21 / 30)	A=5+6*(21/30)_	SIN wurde nicht eingegeben.
◄ ◄ ◄	A=5+6*(21/30)	Rücksetzen des Cursors.
SHFT INS	A=5+6*(21/□30)	Einfügen dreier Leerzeichen für den Befehl SIN.
SHFT INS SHFT INS	A=5+6*(21/□□□30)	
S I N	A=5+6*(21/SIN30)	Eingabe des Befehls.
ENTER	257.	

(3) Es wurde versehentlich 211 anstelle von 21 eingegeben

Eingabe	Anzeige	Anmerkung
"DEG"		
A = 5 + 6 * (	A=5+6*(_	
211 / S I N 30)	A=5+6*(211/SIN30)_	
◄ ◄ ◄ ◄	A=5+6*(211/SIN30)	Rücksetzen des Cursors auf die zu löschende Zahl 1.
◄ ◄ ◄ ◄	A=5+6*(211/SIN30)	
SHFT DEL	A=5+6*(21/SIN30)	Löschen des falschen Zeichens.
ENTER	257.	

(4) Anstelle von 6 wurde 2 eingetastet. Der Fehler wird erst nach Berechnung des Ergebnisses bemerkt.

Eingabe	Anzeige	Anmerkung
"DEG"		
A = 5 + 2 * (	A=5+2*(_	
21 / S I N 30)	A=5+2*(21/SIN30)_	
ENTER	89.	Ergebnis erscheint unglaubwürdig.
◄	A=5+2*(21/SIN 30)	Rückruf der Eingabe.
► ► ► ►	A=5+2*(21/SIN 30)	Verschieben des Cursors.
6	A=5+6*(21/SIN 30)	Überschreiben der 2 mit 6.
ENTER	257.	

(5) Da SIN 0 anstelle von SIN 30 eingegeben wurde, führt die Ausführung der Rechnung wegen der Division durch Null zur Fehlermeldung mit dem Code 1.

Eingabe	Anzeige	Anmerkung
"DEG" [A] [=] 5 [+] 6 [*] [(]	A=5+6* (_	
21 [/] [S] [I] [N] 0 [)]	A=5+6* (21/SIN0) _	
[ENTER]	1	Fehlermeldung.
[►]	A=5+6* (21/SIN 0)	Die Eingabe wird zurückgerufen.
[◄] [SHFT] [INS]	A=5+6* (21/SIN □0)	Eine 3 wird eingefügt.
3	A=5+6* (21/SIN 30)	
[ENTER]	257.	

Bei der Speicherung eines Ausdrucks mit [ENTER] werden die Zeichen vom Rechner bereits logisch codiert. Die Zeichen SIN sind dann keine willkürlichen Buchstaben mehr sondern der Befehl sin. Demgemäß wird nach dem Rückruf der Eingabe die Folge SIN als Einheit aufgefaßt und vom Cursor in einem Schritt übersprungen.

13. RANGORDNUNG DER OPERATOREN

Die im PC-1212 konsequent angewandte Algebraische Eingabelogik mit Hierarchie ermöglicht es, alle Probleme so in den Rechner einzutasten, wie es der Ansatz vorschreibt. Da auch die Klammern entsprechend der Vorlage gesetzt werden und geschachtelt werden dürfen, kann man ohne Kenntnis der rechnerinternen Vorrangordnung jede Aufgabe meistern. Trotzdem soll in diesem Abschnitt die Hierarchie des Rechners klargelegt werden, da der routinierte Anwender dann eine Reihe von Problemen vereinfachen und elegante Lösungswege finden kann.
Eine besondere Vereinfachung der Eingabe bietet der Rechner bei Multiplikationen mit Speicherinhalten und Pi. Hier darf nämlich das Multiplikationskreuz weggelassen werden.

Beispiel: A B C = A x B x C
2 A = 2 x A
5 B π = 5 x B x π

Der Rechner geht bei der Abarbeitung eines eingetasteten, im Eingaberegister stehenden Ausdrucks nach der nachfolgenden Vorrangordnung vor. Funktionen aus einer Gruppe mit der niedrigeren Ziffer werden vorrangig vor Funktionen mit der größeren Gruppenziffer ausgeführt.

(1) Abruf von Pi und Abruf der Speicher A bis Z.
(2) Abruf der indizierten Speicher der allgemeinen Form A ( ).
(3) Allgemeine Exponentialfunktion *wenn* eine Multiplikation ohne [*] -Befehl vorausgeht.
Beispiel: $2 A \wedge 3 \triangleq 2 * A^3$
(4) Multiplikation ohne [*] -Befehl.
Beispiel: 2 A; π B; A B
(5) Die Funktionen SIN, COS, TAN, ASN, ACS, ATN, LN, LOG, EXP, DMS, DEG, INT, ABS, SGN, $\sqrt{}$.
(6) Allgemeine Exponentialfunktion *ohne* vorausgehende Multiplikation ohne [*] -Befehl.
(7) Vorzeichen + und −.
(8) Multiplikation * und Division /.
(9) Addition + und Subtraktion −.
(10) Vergleichsoperatoren =, >, <, >=, <=, <>.

Ausdrücke, die in Klammern eingeschlossen sind, werden vorrangig behandelt. Bei geschachtelten Klammern wird das innerste Klammerpaar vor allen anderen berücksichtigt.

Werden Funktionen mit gleicher Rangordnung aneinandergereiht, so arbeitet sie der Rechner von rechts nach links ab.

Beispiel: LN ABS A = ln |A|, EXP $\sqrt{\ }$ 8 = exp ($\sqrt{8}$);
3 ∧ 4 ∧ 2 = 3 exp (4 exp 2).

14. SPEICHERORGANISATION

Will man den Rechner optimal nutzen, sollte man sich etwas mit der Organisation der Register befassen. Register sind die Rechenwerke und Speicher für Zahlen und Befehle. Ihr Rechner besitzt folgende Register:

14.1 Eingaberegister

Der Computer speichert die eingegebenen Daten, also Zahlen, Buchstaben und Befehle im Eingaberegister. Dieses Register hat eine Speicherkapazität von 80 Zeichen, man darf also bis zu 80 Tasten bedienen. Mit der Taste [ENTER] werden die Daten ihrem Sinn entsprechend in rechnerinterne Codes umgewandelt, welche die Rechenregister zur Ermittlung des Ergebnisses verwenden. Nach Eingabe des 80. Zeichens blinkt der Cursor auf diesem Zeichen. Versucht man, mehr als 80 Tasten einschließlich der [ENTER] -Taste zu bedinenen, überschreibt der 81. Schritt den 80. und die Anzeige wird nicht mehr nach links gerollt.

14.2 Rechenregister X

Dieses Register speichert die zuletzt eingegebene oder berechnete Zahl mit voller Stellenanzahl.

14.3 Pufferregister

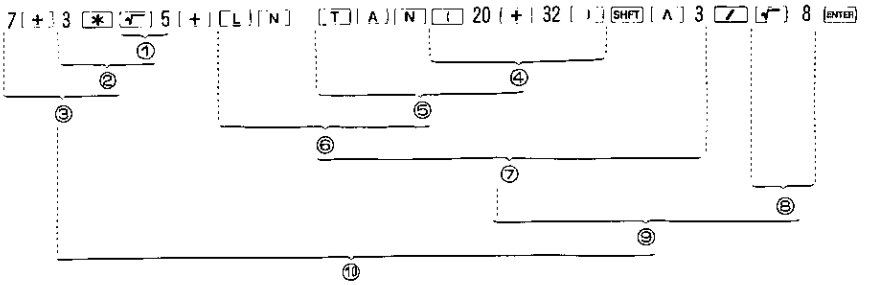
Da der Rechner gemäß den Regeln der Algebra rechnet, ist es normalerweise notwendig, bei der Ausführung einer Rechnung Zahlen und Funktionsbefehle zwischenzuspeichern, zu puffern, bis sie verwendet werden dürfen. Das sieht man an dem einfachen Beispiel

1 + 2 * 3 = 7

Zunächst müssen die Zahl 1 und der Befehl + gespeichert werden, da ja erst die Multiplikation auszuführen ist. Auch 2 und * sind zu speichern, es könnte sich ja der in der Hierarchie höherstehende Befehl ∧ anschließen. Die zuletzt eingegebene Zahl steht jeweils im X-Register; im Beispiel ist das die 3. Es werden also 2 Pufferspeicher für die Zahlen 1 und 2 sowie 2 Pufferspeicher für die Funktionen + und * benötigt. Der Befehl [ENTER] ruft nun die Pufferregister entsprechend der Hierarchie ab. Zunächst wird * und 2 aus den Registern geholt und der Zwischenwert 6 berechnet. Dann wird der als erstes gespeicherte Befehl + und die erste Zahl 1 abgerufen und das Endergebnis 7 berechnet. Können innerhalb einer Befehlsfolge Teilergebnisse berechnet oder Befehle ausgeführt werden, so geschieht das sofort. Damit vermeidet man unnötige Speicherbelegung.

Der Rechner besitzt 16 Funktions-Pufferregister und 8 Zahlen-Pufferregister. Jedes dieser Register kann eine komplette Zahl entsprechend dem X-Register speichern. Werden mehr als 16 Funktionen oder mehr als 8 Zahlen gespeichert, erfolgt Fehlermeldung.

Die Reihenfolge der Ausführung der Rechenbefehle, gekennzeichnet durch die Zahlen in den Kreisen, bei einem umfangreichen Beispiel zeigt das untenstehende Diagramm.



Die Funktion der Pufferspeicher und das Schieben der Daten von einem Speicher in den nächsten soll am Beispiel $1.2 + A * (3.5 + SIN\ B) \wedge A\ (25)$ mit $A = 2.4$, $B = 30^{\circ}$ und $A(25) = 3$ gezeigt werden.

Befehl	X-Register	Zahlen-Pufferspeicher				Funktions-Pufferspeicher					
		Nr. 1	Nr. 2	Nr. 3		Nr. 1	Nr. 2	Nr. 3	Nr. 4	Nr. 5	
"DEG"											
1.2	1.2										
+	1.2	1.2				+					
A	2.4	1.2				+					
*	2.4	2.4	1.2			*	+				
(	2.4	2.4	1.2			(	*	+			
3.5	3.5	2.4	1.2			(	*	+			
+	3.5	3.5	2.4	1.2		+	(	*	+		
SIN	3.5	3.5	2.4	1.2		SIN	+	(	*	+	
B	30	3.5	2.4	1.2		SIN	+	(	*	+	
)	0.5	3.5	2.4	1.2		+	(	*	+		
	4	2.4	1.2			*	+				
^	4	4	2.4	1.2		^	*	+			
A(	4	4	2.4	1.2		A(	^	*	+		
25	25	4	2.4	1.2		A(	^	*	+		
)	3	4	2.4	1.2		^	*	+			
ENTER	64	2.4	1.2			*	+				
	153.6	1.2				+					
	154.8										

Wie man an dem Beispiel sieht, benötigen einfache Konstante keinen Platz im Funktions-Pufferspeicher; indizierte Konstante dagegen belegen einen Platz. Wenn es die Kapazität des Pufferspeichers erlaubt, können bis zu 15 Klammerebenen verwendet werden.

In den Fällen, in denen man auf die Bedienung der Multiplikationstaste verzichten kann, also bei Multiplikationen vom Typ Zahl * Speicher, Pi * Speicher, Zahl * Pi und Speicher * Speicher, setzt der Rechner intern automatisch den *-Befehl und speichert ihn im Funktions-Pufferregister. Da nun jede Multiplikation einen Funktions-Pufferspeicher belegt, kann man entsprechend der Kapazität des Pufferregisters höchstens 8 Multiplikationen ohne *-Befehl aneinanderreihen.

Beispiel: Das Produkt 2 ABC soll ohne Verwendung der Taste [*] berechnet werden. Die Konstanten seien A = 3, B = 5 und C = 7.

Befehl	X-Register	Zahlen-Pufferspeicher			Funktions-Pufferspeicher		
2	2						
A	3	2			*		
B	5	3	2		*	*	
C	7	5	3	2	*	*	*
ENTER	35	3	2		*	*	
	105	2			*		
	210						

14.4 Konstantenregister

Diese Register nehmen komplette Zahlenwerte einschließlich Vorzeichen und Exponent oder Taxte bis zu 7 Buchstaben auf. Ihr Inhalt wird bei der Berechnung des Ergebnisses nicht beeinflußt.

14.5 Funktionsregister

Diese internen Register berechnen die Funktionswerte beim Abruf mathematischer Funktionen durch Reihenentwicklung oder andere Näherungsmethoden. Sie arbeiten unabhängig von den übrigen Registern.

14.6 Programmregister

Der PC-1212 hat ein programmregister mit 1424 Schritten. Ein Schritt nimmt dabei nicht nur ein Zeichen sondern einen kompletten Befehl, etwa SIN oder INPUT auf.

14.7 RESERVE-Register

Das RESERVE-Register nimmt die den RESERVABLE KEYS zugeordneten Befehle und Programme auf. Es hat eine Kapazität von 48 Schritten.

RECHENMÖGLICHKEIT

Funktionen	Dynamischer Bereich
$y \wedge x$ (y^x)	$0 \leqq y$, $-1 \times 10^{100} < x \log y < 100$ (Hier, $y \wedge x = 0$ bei $y = 0$)
SIN x COS x TAN x	DEG: $\lvert x\rvert < 1 \times 10^{10}$ RAD: $\lvert x\rvert < \frac{\pi}{180} \times 10^{10}$ GRAD: $\lvert x\rvert < \frac{10}{9} \times 10^{10}$ In TAN x jedoch werden die folgenden Fälle ausgeschlossen. DEG: $\lvert x\rvert = 90\ (2n-1)$ RAD: $\lvert x\rvert = \frac{\pi}{2}\ (2n-1)$ GRAD: $\lvert x\rvert = 100\ (2n-1)$ (n = ganze Zahl)
SIN^{-1} x COS^{-1} x	$-1 \leqq x \leqq 1$
TAN^{-1} x	$\lvert x\rvert < 1 \times 10^{100}$
LN x LOG x	$1 \times 10^{-99} \leqq x < 1 \times 10^{100}$
EXP x	$-1 \times 10^{100} < x \leqq 230{,}2585092$
$\sqrt{x}$	$0 \leqq x < 1 \times 10^{100}$

Andere Funktionen als die obenstehenden können nur berechnet werden, wenn x innerhalb des folgenden Bereichs steht.

$1 \times 10^{-99} \leqq \lvert x\rvert < 1 \times 10^{100}$ und 0

V PROGRAMMIEREN IN BASIC

Der PC-1212 verwendet die weitverbreitete Programmiersprache BASIC. Das Wort BASIC steht für Beginners Allpurpose Symbolic Instruction Code. Der große Vorteil dieser Sprache liegt neben ihrer einfachen Erlernbarkeit darin, daß sie den Dialogbetrieb, die direkte Korrespondenz zwischen Rechner und Anwender, gestattet.

1. AUFBAU DER SPRACHE BASIC

Wie jede Fremdsprache besitzt BASIC einen begrenzten *Zeichenvorrat*. Er umfaßt die Großbuchstaben A bis Z, die Ziffern 1 bis 9 und eine Reihe von Sonderzeichen wie Symbole für Rechenoperationen, Vergleichsoperatoren oder auch Zeichen, die nur für die Verwendung in Kommentaren gedacht sind wie etwa % oder ¥. Um Verwechslungen zu vermeiden, wird die Ziffer Null als 0 dargestellt; der Buchstabe O dagegen erhält das Symbol Ø.
Aus diesem Zeichenvorrat setzt man die *Sprachelemente* von BASIC zusammen. Auf den ersten Blick erscheint es umständlich, daß man im Gegensatz zu den üblichen Taschenrechnern Funktionen wie sin oder lg mit mehreren Tastenbedienungen, also [S] [I] [N] oder [L] [O] [G] eingeben muß. Wegen der vielen Möglichkeiten, die BASIC bietet, müßte der Rechner aber über 100 Tasten besitzen, wollte man jeder Funktion eine Taste zuordnen. Das würde viel mehr Sucharbeit erfordern als die einfache Eingabe mit der von der Schreibmaschine gewohnten ASCII-Tastatur.

BASIC kennt folgende Sprachelemente: (Beispiele in Klammern):

Konstante (14,123; 3×10^{-10})
Variable (A; B; A(22); D$)
Mathematische Funktionen (SIN; LOG)
Operationszeichen (+; /)
Steueranweisungen (GOTO; FOR)
Eingabeanweisungen (LET; INPUT)
Ausgabeanweisungen (PRINT; USING)
Kommandoanweisungen (RUN; DEBUG)

Die Sprachelemente werden wie Worte zu *Programmsätzen*, Befehlsfolgen für den Rechner, zusammengefügt.

Die Programmsätze ordnet man in *Zeilen* an, die mit einer Zeilennummer zu versehen sind. In einer Zeile können auch mehrere Programmsätze getrennt durch den Doppelpunkt [SHIFT] [:] stehen.

Das Zusammenfügen der Sprachelemente zu Programmzeilen und schließlich zum vollständigen Programm soll am Beispiel des Satzes von Pythagoras im folgenden Kapitel gezeigt werden. Dabei werden zunächst mehrere Sparachelemente verwendet, die erst in den folgenden Kapiteln erklärt werden. Das Beispiel soll jedoch nur den prinzipiellen Aufbau eines Programms zeigen.

2. PROGRAMMAUFBAU

In einem rechtwinkligen Dreieck mit den Katheten a und b berechnet man die Länge der Hypotenuse c mit dem Satz des Pythagoras $c = \sqrt{a^2 + b^2}$. Die Katheten a und b können beliebige Werte annehmen, sie sind also Variable. Beim nichtprogrammierten Rechnen muß man den Variablen a und b bestimmte Werte zuordnen; im Beispiel sollen a = 3 und b = 4 sein. Diese Werte werden in die Speicher A und B eingelesen. Ohne Programmunterstützung ermittelt man c auf folgende Weise:

Eingabe	Anzeige	Anmerkung
"RUN"	>	Wahl der Betriebsart RUN
[A] [=] 3 [ENTER]	3.	Eingabe von a
[B] [=] 4 [ENTER]	4.	Eingabe von b
[C] [=] [√] [(] [A] [*] [A]	C=√(A*A_	
[+] [B] [*] [B] [)]	C=√(A*A+B*B)_	
[ENTER]	5.	$\sqrt{A^2+B^2}$

Möchte man für viele Wertepaare (a, b) die Gleichung auswerten, so muß man beim manuellen Rechnen für jedes Wertepaar den gesamten Rechenablauf eintasten. Viel einfacher wäre es, die jedem Rechengang gleiche Grundstruktur einmal zu programmieren um dann nur mehr die Wertepaare (a, b) einzutasten. Genau das ist beim PC-1212 in der Betriebsart PRO möglich. Das Programm 1, "Satz des Pythagoras" lautet wie folgt:

PROGRAMM 1

Programmzeile	Anmerkung
10 : INPUT A , B	Eingabeanweisung
20 : C=√(A*A+B*B)	Mathematischer Ausdruck
30 : PRINT C	Ausgabeanweisung
40 : END	Programmbeendungsanweisung

Das Programm wird also ähnlich geschrieben, wie der Ansatz beim manuellen Rechnen unter Verwendung der Speicher. Es enthält vier Grundelemente, die in jedem Programm benötigt werden:

(1) Die Eingabeanweisung (INPUT)
Die Eingabeanweisung bestimmt Anzahl und Namen der Variablen (A, B). Gleichzeitig sorgt der Rechner dafür, daß bei der Programmausführung im Dialogbetrieb die Zahlenwerte für diese Variablen abgefragt werden.

(2) Der mathematische Ausdruck (C = √(A*A + B*B))
Dieser Ausdruck verknüpft die Variablen A und B zum Rechenergebnis C.

(3) Die Ausgabeanweisung (PRINT)
Die Ausgabeanweisung veranlaßt die Anzeige des Rechenergebnisses.

(4) Die Programmbeendungsanweisung (END)
Am Schluß eines Programms teilt man dem Rechner mit dem Befehl END mit, daß keine weiteren Instruktionen folgen.

Neben diesen Grundelementen gibt es noch viele weitere Programmelemente, die in den folgenden Kapiteln erläutert werden. Besonders vorteilhaft ist die Möglichkeit, die Bedeutung der Variablen mit Hilfe eines Textes zu kommentieren.

Hat man ein Programm geschrieben, so wird es in Betriebsart PRO eingetastet. Bei Programm 1 geschieht das auf folgende Weise:

Eingabe	Anzeige	Anmerkung
"PRO"	>	(1)
[N] [E] [W] [ENTER]	>	(2)
10 [I] [N] [P] [U] [T]	10 INPUT_	(3)
[A] [SHIFT] [,] [B]	10 INPUTA , B_	(4)
[ENTER]	10 : INPUT A , B	(5)
20 [C] [=] [√] [(] [A]	20C=√(A_	
[*] [A] [+] [B]	20C=√(A*A+B_	
[*] [B] [)]	20C=√(A*A+B*B) _	(6)
[ENTER]	20 : C=√(A*A+B*B)	(7)
30 [P] [R] [I] [N] [T]	30PRINT_	
[C]	30PRINTC_	
[ENTER]	30 : PRINT C	(8)
40 [E] [N] [D]	40END_	
[ENTER]	40 : END	(9)

Anmerkungen:

(1) Wahl der Betriebsart Programmieren. Die Symbole PRO (Betriebsart) und > (Bereitschaft) müssen in der Anzeige stehen.

(2) Vor dem Programmieren muß man einen Speicherplatz für das neue Programm suchen. Hier geschieht das einfach dadurch, daß alle alten Programmzeilen gelöscht werden.

(3) Nach Eingabe der Zeilennummer darf kein Doppelpunkt gesetzt werden. Die Eingabeanweisung wird direkt im Anschluß an die Zeilennummer geschrieben.

(4) Die beiden Variablen A und B werden festgelegt.

(5) Nachdem die komplette Zeile 10 ins Eingaberegister eingetastet und überprüft ist, wird sie mit dem Befehl [ENTER] ins Programmregister geschoben. Dabei werden automatisch der Doppelpunkt nach der Zeilennummer und Leerzeichen zwischen den Befehlen eingefügt.

(6) In Zeile 20 wird der mathematische Ausdruck ins Eingaberegister eingetastet.

(7) [ENTER] schiebt Zeile 20 vom Eingaberegister ins Programmregister.

(8) Die Ausgabeanweisung Zeile 30 kommt ins Programmregister.

(9) Das Programm 1 wird mit dem Befehl END abgeschlossen. In der nächsten Zeile kann ein weiteres Programm gespeichert werden.

Um die programmierte Instruktion zu verwenden, schaltet man durch dreimaliges Bedienen der Taste [MODE] in die Betriebsart RUN um. Die Programmausführung verläuft bei Programm 1 nach folgendem Schema:

Eingabe	Anzeige	Anmerkung
"RUN"	>	Betriebsbereitschaft in Betriebsart RUN.
R U N	RUN _	Die Kommandoanweisung für die Ausführung des Programms lautet RUN.
ENTER	?	Der Rechner fragt mit dem Zeichen ? nach dem Zahlenwert der ersten Variablen.
3	3	Die Variable A erhält den Wert 3; 3 wird also in Konstantenspeicher A geschrieben.
ENTER	?	Abfragen des Zahlenwertes der zweiten Variablen.
4	4	Die Variable B erhält den Zahlenwert 4.
ENTER	5	Rechenergebnis; es steht im Speicher C.
ENTER	>	Der Rechner ist für einen neuen Rechengang beginnend mit dem Befehl RUN bereit.

Dieses einfache Programm zeigt die Fähigkeit des Rechners, eine Rechenroutine mit vielen verschiedenen Zahlenwerten abzuarbeiten. Es hat allerdings noch einige Schönheitsfehler. Einmal muß man jeden Programmdurchlauf mit den Tasten R U N starten, zum anderen sagt zwar das Fragezeichen, daß eine Variable einzugeben ist, nicht aber welche. Man muß daher neben dem Rechner Aufzeichnungen benutzen. Das folgende modifizierte Programm des Satzes von Pythagoras vermeidet diese Nachteile und zeigt, wie leistungsfähig BASIC ist.

```
10:  PRINT "SATZ DES PYTHAGORAS"
20:  INPUT "KATHETE A="; A
30:  INPUT "KATHETE B="; B
40:  C = √ (A * A + B * B)
50:  PRINT "HYPOTENUSE C="; C
60:  GOTO 20
70:  END
```

Beim Abarbeiten dieses Programms in Stellung RUN meldet sich der Rechner auf das Kommando RUN mit "SATZ DES PYTHAGORAS" und fragt dann KATHETE A= und KATHETE B=. Nach Berechnung von C kommentiert er sein Rechenergebnis mit "HYPOTENUSE C= Rechenergebnis".

3. PROGRAMMEINGABE

Im Kapitel 2 wurde bereits anhand eines Beispiels die Eingabe eines Programms gezeigt. Allgemein geht man nach folgendem Schema vor:

(1) Einschalten des Rechners.

(2) Wahl der Betriebsart PRO mit der Taste MODE .
In der Anzeige stehen die Symbole PRO und >.

(3) Aufsuchen eines geeigneten Speicherplatzes.

(3.1) Interessieren die alten Programme und Daten nicht mehr, löscht man mit der Kommandoanweisung N E W ENTER . Dadurch werden der ganze Programmspeicher und alle Konstantenspeicher gelöscht. Lediglich die den RESERVABLE KEYS zugeordneten Daten bleiben gespeichert.

(3.2) Möchte man alte Programme erhalten, läßt man das neue Programm mit einer Zeilennummer beginnen, die höher als die der letzten gespeicherten Zeile des alten Programms ist. Steht etwa das END des alten Programms in Zeile 140, so beginnt man das neue mit Programmzeile 150.

(3.3) Soll das Programm einen Programmnamen, etwa "A" oder "B" bekommen, damit es sich schneller auffinden läßt, arbeitet man in Betriebsart DEF. Diese Routine ist in Kapitel V9 beschrieben. Als Programmname kann auch ein Text bis zu 7 Buchstaben dienen, den man in Betriebsart RUN oder DEF anwählt.

(4) Eintasten einer Programmzeile.

(4.1) Wahl der Zeilennummer.
Möglich sind die ganzen Zahlen von 1 bis 999. Empfehlenswert ist es, die Zeilen nicht 1; 2; 3 · · · sondern 5; 10; 15 · · · oder 10; 20; 30 · · · zu numerieren. Dann ist es nämlich möglich, für Programmänderungen neue Zeilen einzufügen. Die Zeilen können in beliebiger Nummernfolge geschrieben werden. Der Rechner ordnet sie automatisch nach aufsteigenden Zeilennummern. Nach dem Eintasten der Zeilennummer wird direkt, also ohne Eintasten eines Doppelpunktes mit dem Programmsatz begonnen.

(4.2) Schreiben mehrerer Programmsätze in einer Zeile.
Es ist möglich, mehrere Programmsätze in einer Zeile zu schreiben. Sie müssen lediglich jeweils durch den Doppelpunkt [SHIFT] [:] getrennt werden. Maximal sind 80 Zeichen, also 80 Tastendrücke einschließlich [ENTER] , pro Zeile möglich.

(4.3) Einlesen der Programmzeile aus dem Eingabespeicher in den Programmspeicher mit der Taste [ENTER] .
Als Quittung dafür, daß die Zeile vom Programmspeicher angenommen wurde, erscheint der Doppelpunkt nach der Programmzeile. Buchstabenfolgen, die eine Anweisung oder eine mathematische Funktion darstellen, werden automatisch durch ein Leerzeichen voneinander getrennt.

(5) Abschluß des Programms
In der letzten Zeile des Programms muß die Steueranweisung END stehen.

4. ÜBERPRÜFUNG DER PROGRAMME

4.1 Überprüfung beim Eintasten

Bereits beim Eintasten einer Programmzeile sollte man den Text auf Schreibfehler überprüfen.

4.2 Überprüfen nach dem Einlesen in den Programmspeicher

Eine zweite Überprüfung der Programmzeile kann nach dem Einlesen in den Programmspeicher mit der Taste [ENTER] erfolgen. Dabei ist die Kontrolle wegen der Trennung der Instruktionen durch Leerzeichen übersichtlicher. Programme mit mehr als 24 Zeichen kann man mit der Taste [►] , eventuell im Schnellauf, in die Anzeige rollen.

4.3 Überprüfung nach Eingabe des gesamten Programms

Wünscht man eine Endkontrolle nach der Eingabe des gesamten Programms, ruft man die gewünschte Zeile mit den Tasten [↓] und [↑] in die Anzeige. Der Druck auf die Taste [↓] rollt die Programmzeilen in Richtung größere Zeilennummer; [↑] dagegen in Richtung kleinere Zeilennummer. Durch längeres Niederdrücken der Tasten kann man die Programmzeilen im Schnellauf abrollen. Die erste eingegebene Zeile sucht die Kommandoanweisung [LIST] [ENTER] oder [CL] [↓] . Die Zeile mit der Nummer n sucht man mit der Kommandoanweisung LIST n [ENTER] . Existiert die gesuchte Zeile nicht, meldet das der Rechner mit dem Fehlercode 2. Ist der gesamte Speicher leer, erscheint nach den Befehlen [↓] , [↑] oder LIST [ENTER] das Bereitschaftssymbol >.

Die Überprüfung eines Programms soll am Beispiel von Programm 1 gezeigt werden. Falls dieses Programm gelöscht oder verändert wurde, muß es entsprechend Kapitel 2 neu eingelesen werden.

Eingabe	Anzeige	Anmerkung
"PRO"	>	
[L][I][S][T][ENTER]	10 : INPUT A , B	Wahl der Betriebsart PRO.
[↓]	20 : C=√(A*A+B*B)	Erste Programmzeile.
[↓]	30 : PRINT C	Abruf mit steigender Zeilennummer.
[↓]	40 : END	
[↑]	30 : PRINT C	Abruf mit fallender Zeilennummer.
[↑]	20 : C=√(A*A+B*B)	
[↑]	10 : INPUT A , B	
[L][I][S][T] 30 [ENTER]	30 : PRINT C	Abruf der Zeile 30.

5. PROGRAMMAUSFÜHRUNG

Den Programmablauf führt man nach folgendem Schema durch:

(1) Einschalten des Rechners.

(2) Wahl der Betriebsart RUN oder DEF mit der Taste [MODE] .

(3) Aufsuchen des Programms.

(3.1) Will man mit dem ersten gespeicherten Programm arbeiten, tastet man die Kommandoanweisung RUN [ENTER] .

(3.2) Das in der Zeile n stehende Programm wählt man mit der Kommandoanweisung RUN n [ENTER] .

(3.3) Das Programm mit dem Programmnamen m wählt man in der Betriebsart DEF mit der Kommandoanweisung [SHIFT] m [ENTER] .

(3.4) Das Programm mit dem Programmnamen "Text" wählt man in der Betriebsart RUN oder DEF mit der Kommandoanweisung RUN "Text".

(4) Wird der Programmlauf zur Variableneingabe bei einer INPUT-Anweisung unterbrochen, erscheint in der Anzeige ein Fragezeichen oder der programmierte Text. In diesem Fall gibt man die den Variablen zuzuordnenden Daten, also eine Zahl oder einen Text bis zu 7 Buchstaben, ein und drückt [ENTER] .

(5) Wird der Programmlauf zum Ablesen eines Ergebnisses bei einer PRINT-Anweisung unterbrochen, setzt man das Programm mit [ENTER] ohne Dateneingabe fort.

(6) Der Programmlauf wird beendet, wenn man die Zeile mit der END-Anweisung erreicht. In der Anzeige erscheint das Symbol > ; der Rechner ist für neue Instruktionen aufnahmebereit.

Beispiel: Die Hypotenuse eines rechtwinkligen Dreiecks soll mit den Kathetenpaaren (A = 12.3; B = 15.7) und (A = 36; B = 27) berechnet werden. Programm 1 ist gespeichert.

Eingabe	Anzeige	Anmerkung
"RUN"	)	Betriebsart RUN (2)
R U N	RUN_	Startkommando (3.1)
ENTER	?	Abfrage von A_1 (4)
12.3	12. 3_	Dateneingabe
ENTER	?	Abfrage von B_1
15.7	15. 7_	Dateneingabe
ENTER	19. 94442278	Ergebnis C_1 (5)
ENTER	)	Programmende (6)
R U N ENTER	?	2. Programmlauf
36 ENTER	?	A_2
27 ENTER	45.	B_2, C_2
ENTER	)	Programmende

6. PROGRAMMKORREKTUR

Findet man bei der Programmüberprüfung oder beim Programmlauf Fehler, korrigiert man sie mit den folgenden Prozeduren:

6.1 Korrektur von Programmteilen

(1) Wahl der Betriebsart PRO.

(2) Wahl der zu korrigierenden Programmzeile mit den Tasten ↑ , ↓ oder der Kommandoanweisung LIST.

(3) Markierung des zu korrigierenden Zeichens durch den Cursor mit Hilfe der Tasten ← und → .

(4) Korrektur durch Überschreiben, Einfügen oder Löschen von Zeichen gemäß Kapitel IV 12.

(5) Die korrigierte Programmzeile steht zunächst nur im Eingaberegister. Sie muß unbedingt wieder mit ENTER ins Programmregister geschoben werden.

Beispiel: Man gibt das Programm 1 absichtlich mit Fehlern ein (linkes Programm) und korrigiert die Zeilen 20 und 30 (rechtes Programm).

```
10 : INPUT A , B
20 : C=√ (A*A+B+B)
30 : PAUSE C
40 : END
```

⟶

```
20 : C=√‾(A*A+B*B)
30 : PRINT C
```

	Eingabe	Anzeige	Anmerkung
	"PRO"	>	Betriebsart PRO (1).
1	[↑][↑]	20 : C=√(A*A+B+B)	Wahl der Programmzeile (2).
2	[→]	20 C=√(A*A+B+B)	
3	[→]..... [→]	20 C=√(A*A+B+B)	Markierung des falschen Zeichens (3).
4	[*]	20 C=√(A*A+B*B)	Überschreiben von + mit *(4).
5	[ENTER]	20 : C=√(A*A+B*B)	Speichern (5).
6	[↑]	30 : PAUSE C	
7	[←]	30 PAUSE C	
8	[P]	30 PC	
9	[R]	30 PR_	
10	[I][N][T][C]	30 PRINTC_	
11	[ENTER]	30 : PRINT C	

Bedient man eine der Tasten [→] oder [←] nach der Wahl der Programmzeile (Druckzeilen 2 und 7), so erscheint der Cursor auf dem *ersten* Zeichen nach der Zeilennummer. Der Doppelpunkt hinter der Zeilennummer verschwindet. Die Zeile steht nicht mehr im Programmspeicher; nach wie vor sind aber Instruktionen wie INPUT, PRINT oder SIN durch Leerzeichen voneinander getrennt. Sie werden bei der Korrektur als *ein* Schritt aufgefaßt, also vom Cursor insgesamt übersprungen oder beim Überschreiben des ersten Zeichens vollständig gelöscht (Druckzeile 8).

Einige Anweisungen lassen sich nach einer Fehlermeldung nicht durch Überschreiben des falschen Buchstaben korrigieren. Hat man z.B. PRINT anstelle von PRINT geschrieben, darf man nicht allein T durch K ersetzen, man muß vielmehr die ganze Anweisung PRINT neu schreiben.

6.2 Einfügen von Programmzeilen

(1) Wahl der Betriebsart PRO.

(2) Eingabe der neuen Zeile beginnend mit einer Zeilennummer, die zwischen den Zeilennummern der Programmsätze liegt, zwischen die man die neue Zeile einfügen möcht. Soll die neue Zeile zwischen die Zeilen 10 und 20 eingefügt werden, darf sie eine beliebige Zeilennummer von 11 bis 19 erhalten.

(3) Der Druck auf die Taste [ENTER] schreibt die neue Zeile automatisch an die richtige Stelle im Programmspeicher.

Beispiel: Das Programm 1 soll durch Einfügen des Programmsatzes PAUSE A, B zwischen Zeile 10 und 20 erweitert werden. Man wählt Programmzeile 15.

```
10 : INPUT A , B
20 : C=√(A*A+B*B)
30 : PRINT C
40 : END
```

15 : PAUSE A , B

Eingabe	Anzeige	Anmerkung
"PRO"	)	Betriebsart PRO (1).
15 [P][A][U][S][E]	15PAUSE_	Die neue Zeile 15 (2).
[A][SHIFT][,][B]	15PAUSEA,B_	
[ENTER]	15 : PAUSE A , B	Eingabe in den Programmspeicher (3).
[CL][I]	10 : INPUT A , B	Überprüfung de ersten Programmsatzes.
[I]	15 : PAUSE A , B	
[I]	20 : C=√ (A∗A+B∗B)	

6.3 Löschen von Programmzeilen

(1) Wahl der Betriebsart PRO.

(2) Eintasten der Nummer der zu löschenden Zeile.

(3) Löschen der gewünschten Zeile mit dem Befehl [ENTER] .

Beispiel: Löschen der im Beispiel des Kapitels 6.2 eingefügten Zeile 15.

Eingabe	Anzeige	Anmerkung
"PRO"	)	Wahl der Betriebsart (1).
15	15_	Wahl der Zeilennummer (2).
[ENTER]	)	Löschen der Zeile 15 (3).
[I]	10 : INPUT A , B	Überprüfung des Programms.
[I]	20 : C=√ (A∗A+B∗B)	

7. FEHLERSUCHE

Die Kommandoanweisung DEBUG erlaubt es, Programme zur Fehlersuche in einzelnen Schritten ablaufen zu lassen. Man verwendet folgende Routine:

(1) Wahl der Betriebsart RUN

(2) Eintasten der Kommandoanweisung DEBUG [ENTER] . Jetzt wird immer nur *eine* Programmzeile abgearbeitet.

(3) Kurzer Druck auf [I] arbeitet das Programm bis zur nächsten Dateneingabe oder zur nächsten Zeile ab. Die Nummer der Zeile, die ausgeführt wurde erscheint anschließend in der Anzeige. Hält man die Taste [I] fest, wird das Programm bis zum Loslassen schnell ausgeführt.

(4) Um festzustellen, welcher Schritt des Programmes als nächster ausgeführt wird, kann man jederzeit den bearbeiteten Programmsatz durch Festhalten der Taste [I] anzeigen. Der Cursor steht auf dem betreffenden Schritt.

(5) Zur Kontrolle der eingegebenen Daten kann man jederzeit die Speicher in der üblichen Weise abfragen.

(6) Im Anschluß an (4), (5) oder Dateneingabe führt man die Fehlersuche mit [I] fort.

(7) Möchte man die Fehlersuche abbrechen und normal weiterrechnen, verwendet man die Kommandoanweisung CONT [ENTER] .

(8) Möchte man mitten in einem Programmlauf auf die Fehlersuche umschalten, unterbricht man den Programmlauf mit der Taste [BREAK] und fährt nach (2) weiter.

Ist Programm 1 noch gespeichert, kann man es gemäß untenstehendem Beispiel schrittweise untersuchen:

Eingabe	Anzeige	Anmerkung
"RUN"	<	Betriebsart RUN (1).
[D] [E] [B] [U] [G] [ENTER]	?	Kommando zur Fehlersuche (2).
36	36_	Dateneingabe.
[↓]	?	Fortführung der Fehlersuche (3).
27	27_	Dateneingabe.
[↓]	10:	Anzeige der Zeilennummer des abgearbeiteten Programmsatzes (3).
Festhalten der Taste [↓]	10: INPUT A,B_	Anzeige des bearbeiteten Programmsatzes (4).
Loslassen der Taste [↓]	>	Die Fehlersuche kann fortgeführt werden.
[↓]	20:	Prüfen der Zeile 20 (6).
[↓]	45.	
[A] [ENTER]	36.	Abfrage des Speicherinhalts (5).
[B] [ENTER]	27.	
[↓]	30:	Prüfen der Zeile 30 (6).
[↓]	>	Programmende.

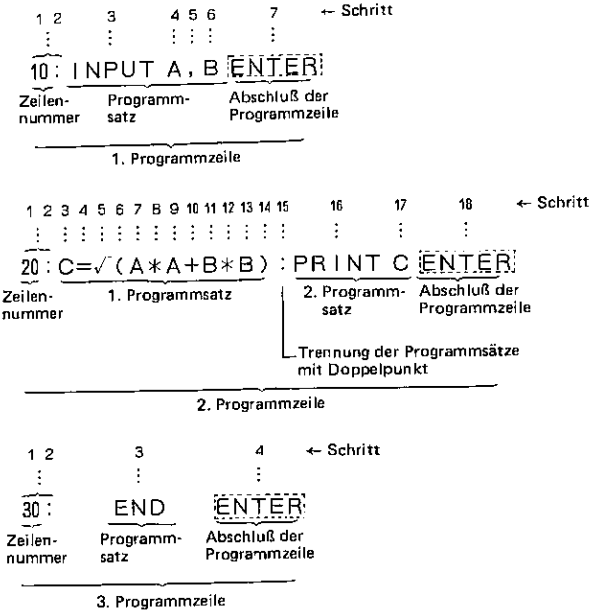
8. ORGANISATION DES PROGRAMMSPEICHERS

8.1 Eingaberegister

Bevor eine Programmzeile mit [ENTER] in den Programmspeicher gestellt wird, muß man sie im Eingaberegister zwischenspeichern. Das Eingaberegister hat 80 Speicherplätze, jeder Tastendruck, auch [ENTER], belegt einen solchen Platz. Man darf daher nur maximal 79 Zeichen schreiben, der 80. Platz muß für [ENTER] frei bleiben. Alle Instruktionen belegen soviele Plätze wie sie Buchstaben bestizen, INPUT also 5 und SIN 3 Eingabe-Speicherplätze.

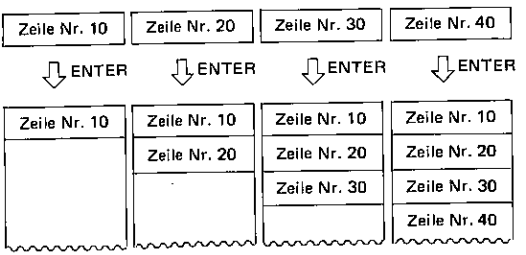
8.2 Programmregister

Um Speicherplätze zu sparen, werden die Instruktionen des Eingabespeichers umcodiert, wenn sie mit [ENTER] in den Programmspeicher geschoben werden. Diese Umcodierung bewirkt, daß im Programmspeicher eine Anweisung oder ein mathematischer Funktionsbefehl nur *einen* Programmspeicherplatz belegt, auch dann, wenn diese Instruktion aus bis zu 6 Buchstaben besteht. Die Zeilennummern von 1 bis 999 belegen in jedem Fall 2 Programmspeicherschritte. Der Doppelpunkt nach der Zeilennummer steht nicht im Programmspeicher. Er darf nicht eingetastet werden und wird als Kontrolle dafür verwendet, daß die Eingabe umcodiert ist und im Programmspeicher steht. Am Beispiel des modifizierten Programms 1 sollen Zeilenaufbau und Programmspeicherbelegung gezeigt werden.

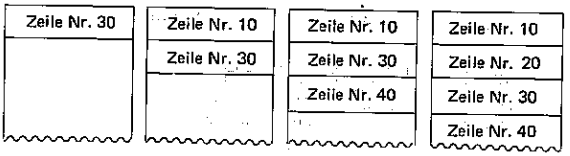

1 2 3 4 5 6 7 ← Schritt
10 : INPUT A , B ENTER
Zeilen-nummer
Programm-satz
Abschluß der Programmzeile
1. Programmzeile
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 ← Schritt
20 : C=√(A*A+B*B) : PRINT C ENTER
Zeilen-nummer
1. Programmsatz
2. Programm-satz
Abschluß der Programmzeile
Trennung der Programmsätze mit Doppelpunkt
2. Programmzeile
1 2 3 4 ← Schritt
30 : END ENTER
Zeilen-nummer
Programm-satz
Abschluß der Programmzeile
3. Programmzeile

8.3 Anordnung der Programmzeilen im Programmregister

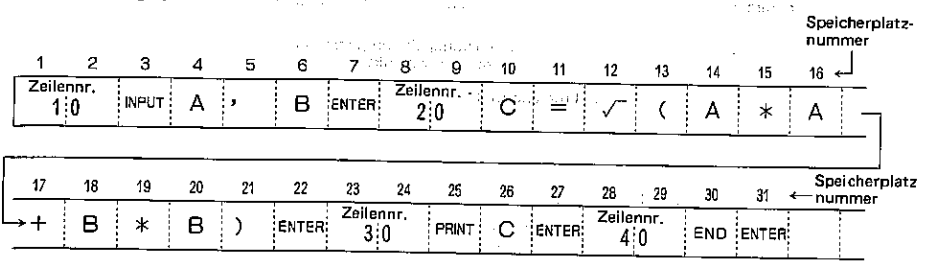
Das Einlesen mehrerer Programmzeilen in den Programmspeicher läuft nach folgendem Schema ab:



Auch wenn die nicht in aufsteigender Zeilennummernfolge eingegeben werden, ordnet sie der Rechner richtig ein. Ist etwa die Eingabefolge der Zeilennummern 30 – 10 – 40 – 20, werden sie nach folgendem Schema in den Programmspeicher eingeordnet:



Das Programmregister selbst hat nicht etwa Teilspeicherbereiche definierter Länge für jede Zeile. Vielmehr speichert er aufeinanderfolgend Schritt für Schritt und setzt die neue Zeilennummer direkt anschließend an den letzten Schritt der vorangehenden Zeile. Dadurch bleiben keine Speicherplätze ungenutzt. Das orginale Programm 1 wird wie folgt im Programmregister gespeichert:



8.4 Abfrage der verfügbaren Programmschritte und Speicherplätze

Möchte man nach der Programmeingabe wissen, wieviele Speicherplätze oder Programmschritte noch frei sind, gibt man in einer beliebigen Betriebsart MEM [ENTER] ein. Der Rechner meldet sich mit m STEPS (Programmschritte) n MEMORIES (Datenspeicher).

9. PROGRAMMNAMEN (LABEL)

Sind mehrere Programme gleichzeitig gespeichert, ist es normalerweise notwendig, sich die Zeilennummern der Programmanfänge zu notieren und die Programme mit der Eingabe

[R] [U] [N] Zeilennummer [ENTER]

in Betriebsart RUN abzurufen. Da das recht umständlich ist, kann man Programme mit Namen versehen und direkt über ihren Namen abrufen. Als Namen dienen die 18 am Tastenfeld besonders markierten Zeichen der RESERVABLE KEYS, nämlich

[A], [S], [D], [F], [G], [H], [J], [K], [L], [=],
[Z], [X], [C], [V], [B], [N], [M], [SPC].

9.1 Benennen des Programms

Als *ersten* Programmsatz eines neuen Programms schreibt man in der Betriebsart PRO den Programmnamen in der allgemeinen Form

[SHFT] [↑↑] Programmname [SHFT] [↑↑]

Steht dieser Satz nicht alleine in einer Zeile sondern wie meist üblich zusammen mit dem nächsten Satz, so darf man ausnahmsweise den Doppelpunkt nach dem ersten Satz weglassen.

9.2 Kontrolle des benannten Programms

Zur Kontrolle ruft man die erste Zeile eines benannten Programms mit der allgemeinen Form

[L] [I] [S] [T] [SHFT] [↑↑] Programmname [SHFT] [↑↑] [ENTER]

oder einfacher

[L] [I] [S] [T] [SHFT] [↑↑] Programmname [ENTER]

in die Anzeige.

9.3 Abruf des benannten Programms

Für den Abruf und die Abarbeitung des benannten Programms ist die Betriebsart DEF zu wählen. Die allgemeine Form des Programmaufrufs lautet:

[SHFT] Programmname

Haben mehrere Programme den gleichen Namen, so wählt der Rechner das Programm mit der niedrigsten Zeilennummer. Sucht man einen Programmnamen, der nicht im Programmregister steht, führt das zur Fehlermeldung mit dem Code 2.

Das Arbeiten mit Programmnamen soll am Beispiel PROGRAMM 2 gezeigt werden.

PROGRAMM 2

Programmzeile	Anmerkung
10 : "A" : INPUT A , B 20 : C=√(A∗A+B∗B) 30 : PRINT C 40 : END	Programmname A $C=\sqrt{a^2+b^2}$: Satz des Pythagoras
50 : "S" : INPUT D 60 : E=4/3∗π∗D^3 70 : PRINT E 80 : END	Programmname S $V=\frac{4}{3}\pi r^3$: Kugelvolumen
90 : " " : INPUT F , G , H 100 : I=√(F∗F+G∗G−2∗F∗G∗COS H) 110 : PRINT I 120 : END	Programmname SPC (Leerzeichen) $C=\sqrt{a^2+b^2-2ab\cos\theta}$: Kosinussatz

Die Programmeingabe in Betriebsart PRO zeigt das folgende Diagramm

Eingabe	Anmerkung
"PRO" [N] [E] [W] [ENTER]	Betriebsart PRO. Speicherlöschung.
10 [SHIFT] [II] [A] [SHIFT] [II] [SHIFT] [:] [I] [N] [P] [U] [T] [A] [SHIFT] [,] [B] [ENTER] 20 [C] [=] [√] [(] [A] [*] [A] [+] [B] [*] [B] [)] [ENTER] 30 [P] [R] [I] [N] [T] [C] [ENTER] 40 [E] [N] [D] [ENTER]	Programmname A; der Doppelpunkt darf fehlen.
50 [SHIFT] [II] [S] [SHIFT] [II] [SHIFT] [:] [I] [N] [P] [U] [T] [D] [ENTER] 60 [E] [=] [4] [/] [3] [*] [SHIFT] [π] [*] [D] [SHIFT] [^] [3] [ENTER] 70 [P] [R] [I] [N] [T] [E] [ENTER] 80 [E] [N] [D] [ENTER]	Programmname S [*] dürfte weggelassen werden.
90 [SHIFT] [II] [SPC] [SHIFT] [II] [SHIFT] [:] [I] [N] [P] [U] [T] [F] [SHIFT] [,] [G] [SHIFT] [,] [H] [ENTER] 100 [I] [=] [√] [(] [F] [*] [F] [+] [G] [*] [G] [−] [2] [*] [F] [*] [G] [*] [C] [O] [S] [H] [)] [ENTER] 110 [P] [R] [I] [N] [T] [I] [ENTER] 120 [E] [N] [D] [ENTER]	Programmname SPC (Leerzeichen)

Zum Abruf der benannten Programme muß auf die Betriebsart DEF geschaltet werden. In dieser Betriebsart kann man ähnlich wie in der Betriebsart RUN Programme abarbeiten und auch manuell rechnen. Nur die in Betriebsart RUN Programme abarbeiten und auch manuell rechnen. Nur die in Betriebsart RESERVE den Tasten zugeordneten Ausdrücke können nicht abgerufen werden.

Eingabe		Anzeige	Anmerkung
"DEF"		⟨	Betriebsart DEF
	[SHIFT] [A]	?	Abruf von Programm A
(A = 4)	4 [ENTER]	?	
(B = 3)	3 [ENTER]	5.	Ergebnis
	[ENTER]	⟩	Ende von Programm A
	[SHIFT] [S]	?	Abruf von Programm S
(D = 2)	2 [ENTER]	33. 51032164	Ergebnis
	[ENTER]	⟩	Ende von Programm A
[D] [E] [G] [R] [E] [E] [ENTER]		⟩	Winkeleinheit Grad (DEG)
	[SHIFT] [SPC]	?	Abruf von Programm SPC
(F = 12)	12 [ENTER]	?	
(G = 14)	14 [ENTER]	?	
(H = 30)	30 [ENTER]	7. 001104508	Ergebnis
	[ENTER]	⟩	Ende von Programm SPC

9.4 Texte als Programmnamen

Als Programmnamen dürfen auch Texte bis zu 7 Buchstaben in Anführungszeichen verwendet werden, z.B. 10: "PYT": INPUT A, B. Derartige Programme sucht man in der Betriebsart RUN mit dem Befehl RUN "Text" [ENTER] .

VI ZUORDNUG VON AUSDRÜCKEN ZU TASTEN

In der Betriebsart RESERVE kann man den 18 auf dem Tastenfeld als RESERVABLE KEYS bezeichneten Tasten

[A], [S], [D], [F], [G], [H], [J], [K], [L], [=], [Z], [X],
[C], [V], [B], [N], [M] und [SPC]

beliebige mathematische Ausdrücke oder Programmieranweisungen zuordnen. Der Rechner besitzt für derartige "RESERVE-Ausdrücke" einen eigenen RESERVE-Speicher mit insgesamt 48 Schritten. Es besteht also keine Verbindung mit Programmen oder Datenspeichern gleichen Namens. Jeder Anwender kann somit spezielle Funktionen, die er häufig benötigt, etwa Hyperbelfunktionen oder Umrechnungen von Maßeinheiten, mit Knopfdruck abrufen. Ordnet man etwa die Ausgabeanweisung [P] [R] [I] [N] [T] der Taste A zu, so kann man den PRINT-Befehl einfach mit [SHIFT] [A] in ein Programm schreiben. Notiert man die selbst definierten Ausdrücke auf der beiliegenden Abdeckschablone, so erhält man den optimal auf die eigenen Bedürfnisse zugeschnittenen Rechner.

Die den RESERVABLE KEYS zugeordneten Ausdrücke sind praktisch kleine Unterprogramme. Man arbeitet daher mit diesen Tasten ähnlich wie beim Programmieren.

1. EINGABE DER RESERVE-AUSDRÜCKE

Um Ausdrücke den RESERVABLE KEYS zuzuordnen, geht man nach folgendem Schema vor:

(1) Wahl der Betriebsart RESERVE mit der Taste [MODE] . In der Anzeige steht > und RESERVE.

(2) Wahl des Namens für den RESERVE-Ausdruck.
Als Name wird der auf den RESERVABLE KEYS stehende Buchstabe verwendet. Sind schon einige RESERVE-Ausdrücke geschrieben, wählt man einen noch nicht verwendeten Namen oder man löscht die alten Ausdrücke.

(2.1) Löschen aller RESERVE-Ausdrücke.
Das gesamte RESERVE-Register löscht man mit

[N] [E] [W] [ENTER]

(2.2) Löschen eines bestimmten RESERVE-Ausdrucks.
Bestimmte RESERVE-Ausdrücke kann man überschreiben oder mit [SPC] schrittweise löschen. Im folgenden Beispiel ist der Taste Z der Ausdruck RUN 50 zugeordnet. Er soll gelöscht werden.

Eingabe	Anzeige	Anmerkung
"RESERVE"	>	Wahl der Betriebsart.
[SHIFT] [Z]	Z : RUN 50	Rückruf des Ausdrucks.
[►]	Z : RUN 50	Abruf des Cursors.
[SPC]	Z : 50	1. Löschschritt.
[SPC] [SPC]	Z : _	Der Ausdruck ist gelöscht.
[ENTER]	>	Beenden des Löschvorgangs.

(3) Eingabe des Ausdrucks.
Die allgemeine Form der Zuordnung eines Ausdrucks zu einer der RESERVABLE KEYS lautet:

[SHIFT] Tastenname Ausdruck [ENTER]

Versucht man mehr als die erlaubten 48 Schritte einzulesen, erscheint nach Bedienen der Taste [ENTER] Fehlermeldung mit dem Fehlercode 4.

Beispiel: Man führe folgende Zuordnungen aus:

"COS" zu [A]
"A*A+B*B" zu [S]
"RUN 130" zu [Z]

PROGRAMM 3

Eingabe	Anzeige	Anmerkung
"RESERVE"		Betriebsart RESERVE.
[N] [E] [W] [ENTER]	>	Löschen des RESERVE-Registers.
[SHIFT] [A]	A : _	Wahl des Namens; der Doppelpunkt wird automatisch gesetzt.
[C] [O] [S]	A : COS_	Ausdruck.
[ENTER]	A : COS	Eingabe ins RESERVE-register.
[SHIFT] [S]	S : _	
[A] [*] [A] [+] [B] [*] [B]	S : A*A+B*B_	
[ENTER]	S : A*A+B*B	
[SHIFT] [Z]	Z : _	
[R] [U] [N] [1] [3] [0]	Z : RUN130_	
[ENTER]	Z : RUN 130	

2. ÜBERPRÜFUNG DER RESERVE-AUSDRÜCKE

Die Überprüfung der gespeicherten Ausdrücke erfolgt in der Betriebsart RESERVE in der allgemeinen Form

[SHIFT] Tastenname

Beispiel: Es sollen die in Programm 3 gespeicherten Ausdrücke überprüft werden.

Eingabe	Anzeige	Anmerkung
"RESERVE"		Betriebsart RESERVE.
[SHIFT] [A]	A : COS	Ausdruck A.
[SHIFT] [S]	S : A*A+B*B	Ausdruck S.
[SHIFT] [D]	D : _	D ist kein Ausdruck zugeordnet.
[SHIFT] [Z]	Z : RUN 130	Ausdruck Z.

3. ABRUF DER RESERVE-AUSDRÜCKE

Der Abruf der gespeicherten Ausdrücke ist in den Betriebsarten RUN und PRO möglich. Die allgemeine Form des Abrufs der RESERVE-ausdrücke lautet:

[SHIFT] Tastenname

Ruft man eine Taste auf, der kein Ausdruck zugeordnet ist, so erscheint der Tastenname in der Anzeige. RESERVE-Ausdrücke können auch direkt hintereinander abgerufen und verknüpft werden.

Beispiel 1: Manuelles Rechnen unter Verwendung der im Programm 3 definierten Ausdrücke. Man berechne cos 60° und $32^2 + 53^2$.

Eingabe	Anzeige	Anmerkung
"RUN" "DEG"		Betriebsart RUN; Winkeleinheit Grad (DEG).
[SHIFT] [A]	COS _	Ausdruck A.
60	COS 60_	
[ENTER]	0. 5	cos 60° =
[A] [=] 32 [SHIFT] [,]	A=32,_	
[B] [=] 53 [ENTER]	53.	
[SHIFT] [S]	A*A+B*B_	Ausdruck S.
[ENTER]	3833.	$32^2 + 53^2$ =

Beispiel 2: Der Kosinussatz $C = \sqrt{a^2 + b^2 - 2ab\cos\gamma}$ soll unter Verwendung der in Programm 3 gespeicherten Ausdrücke beginnend mit Programmzeile 130 programmiert werden.

Eingabe	Anzeige	Anmerkung
"PRO"		Betriebsart PRO.
130 [I] [N] [P] [U] [T]	130 INPUT_	
[A] [SHIFT] [,] [B]	130 INPUTA,B_	A = a, B = b, D = γ.
[SHIFT] [,] [D] [ENTER]	130 : INPUT A,B,D	
140 [C] [=] [√] [(]	140C=√ (_	
[SHIFT] [S]	140C=√ (A*A+B*B_	Einbau von Ausdruck S.
[−] [2] [*] [A] [*]	140C=√ (A*A+B*B−2*A*_	
[B] [*] [SHIFT] [A]	0C=√(A*A+B*B−2*A*B*COS _	Einbau von Ausdruck A.
[D] [)] [ENTER]	140 : C=√ (A*A+B*B−2*A*B*	
150 [P] [R] [I] [N] [T]	150PRINT_	
[C] [ENTER]	150 : PRINT C	
160 [E] [N] [D] [ENTER]	160 : END	Programmende.

Das Programm soll nun mit den Werten a = 10, b = 12 und $\gamma = 60$ ablaufen. Der Programmanfang wird mit Ausdruck Z gewählt.

Eingabe	Anzeige	Anmerkung
"RUN" "DEG"		Betriebsart RUN; Winkeleinheit Grad (DEG).
SHFT Z	RUN130_	Ausdruck Z.
ENTER	?	
(A = 10) 10 ENTER	?	
(B = 12) 12 ENTER	?	
(D = 60) 60 ENTER	11. 13552873	C =

Beispiel 3: Nachträgliches Einfügen von RESERVE-Ausdrücken in Leerstellen der Anzeige. Sind mehrere Plätze frei, wird der Ausdruck nach dem Modus 1 Instruktion ≙ 1 Platz eingelesen. Hat der RESERVE-Ausdruck mehr Instruktionen, als Plätze frei sind, wird die Anzeige soweit nach rechts verschoben, daß keine Information verloren geht.

Eingabe	Anzeige	Anmerkung
"RUN" "DEG"		Betriebsart RUN; Winkeleinheit Grad (DEG).
5 * 6 * 60	5*6*60_	
◄ ◄ SHFT INS	5*6*□60	1 freier Platz.
SHFT A	5*6*COS 60	1 Instruktion.
◄ SHFT INS SHFT INS	5*6*□□COS 60	2 freie Plätze.
SHFT S	5*6*A*A+B*BCOS 60	7 Instruktionen; trotzdem kein Informationsverlust.
*	5*6*A*A+B*B*60	
SHFT INS		
SHFT INS SHFT INS	5*6*A*A+B*B*□□□60	3 freie Plätze.
SHFT A	5*6*A*A+B*B*COS □□60	1 Instruktion; bleiben also 2 Plätze frei.

4. KORREKTUR VON RESERVE-AUSDRÜCKEN

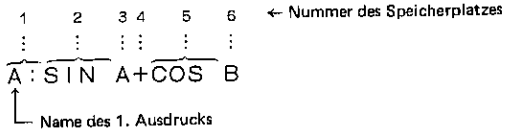
Die gespeicherten Ausdrücke können durch Überschreiben, Löschen und Einfügen korrigiert und verändert werden. Als Beispiel soll das Programm 3 so umgeschrieben werden, daß A den SIN anstelle des COS enthält; S soll LOG A anstelle von A∗A+B∗B und Z der Programmierbefehl RUN 50 anstelle von RUN 130 zugeordnet sein.

Eingabe	Anzeige	Anmerkung
"RESERVE"		Betriebsart RESERVE.
SHIFT A	A:COS	Abruf des Ausdrucks A.
◄	A:COS	Abruf des Cursors.
S I N	A:SIN_	Überschreiben.
ENTER	A:SIN	Eingabe ins RESERVE-Register.
SHIFT S	S:A∗A+B∗B	Abruf des Ausdrucks S.
►	S:A∗A+B∗B	
L O G A	S:LOGAB∗B	Überschreiben.
SPC SPC SPC	S:LOGA _	Löschen der restlichen Zeichen.
ENTER	S:LOG A	
SHIFT Z	Z:RUN 130	Abruf des Ausdrucks Z.
► ►	Z:RUN 130	Positionieren des Cursors.
5 0 SPC	Z:RUN 50 _	Ändern der Zeilennummer.
ENTER	Z:RUN 50	

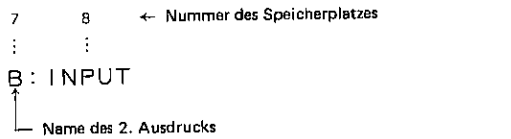
5. ORGANISATION DES RESERVE-REGISTERS

Das RESERVE-Register ist ähnlich dem Programm-Register organisiert. Lediglich steht der Name des gespeicherten Ausdrucks anstelle der Zeilennummer. Dieser Name nimmt nur einen Speicherplatz ein. Insgesamt besitzt das Programmregister 48 Speicherplätze.

Beispiel 1: Der Ausdruck S I N A + C O S B wird Taste A zugeordnet.



Beispiel 2: Im Anschluß an Beispiel 1 ordnet man die INPUT-Anweisung der Taste B zu.



VII VARIABLEN

1. VARIABLENTYPEN

Variablen werden beim Rechner durch Speicher realisiert, denen man unterschiedliche Daten zuordnen kann. Zur leichteren Identifizierung erhalten sie einen Variablennamen als symbolische Adresse eines Speicherplatzes. Mit Variablen lassen sich Gesetzmäßigkeiten unabhängig von ihrem jeweiligen Wert allgemein ausdrücken.
Man unterscheidet zwei Variablentypen: Zahlenvariablen und Zeichenvariablen. Beide Typen können als einfache und als indizierte Variablen geschrieben werden.

1.1 Zahlenvariablen

Einer Zahlenvariablen kann man eine vollständige Zahl bestehend aus zehnstelliger Mantisse, zweistelligem Exponenten und zwei Vorzeichen zuordnen. Daher wurden sie bereits beim manuellen Rechnen als Konstantenspeicher benutzt. Als einfache Zahlenvariablen dienen die 26 Buchstabentasten mit den Variablennamen A bis Z. Indizierte Variablen bestehen aus dem Buchstaben A und einem Index, der beim PC-1212 von 1 bis 204 laufen darf. Der Vorteil dieses Variablentyps ist, daß der Index auch das Ergebnis einer Rechnung oder eine andere Variable sein darf. Ist das Rechenergebnis x keine natürliche Zahl, verwendet der Rechner den ganzzahligen Anteil INT x als Index. Damit ist die indirekte Adressierung, die im Kapitel 5 dieses Teils besprochen wird, möglich.

1.2 Zeichenvariablen (Stringvariablen)

Einer Zeichenvariablen kann man Texte bestehend aus Buchstaben, Zahlen, Sonderzeichen und Leerstellen zuordnen. Bei manueller Eingabe muß dieser Text in Anführungszeichen stehen. Einfache Zeichenvariablen haben die Namen A$ bis Z$; indizierte Zeichenvariablen A$(1) bis A$(204). Man beachte streng den Unterschied zwischen Zahlen- und Zeichenvariablen: A = TEE ordnet A den Zahlenwert T * E * E zu wobei für T und E die Zahlen in den Speichern T und E genommen werden. A$ = "TEE" dagegen ordnet der Zeichenvariablen A$ das Wort TEE zu.

2. ORGANISATION DER VARIABLENSPEICHER

Der Rechner verfügt über zwei Sorten von Variablenspeicher. Der Basisspeicher besitzt 26 Speicherplätze. Er ist unabhängig von den Programmen immer verfügbar. Benötigt man mehr als 26 Variable, kann man Programmspeicher in Variablenspeicher umwandeln. Es werden 8 Programmschritte für einen Variablenspeicherplatz benötigt. Beide Speichertypen können entweder eine vollständige Zahl oder einen Text bis zu 7 Zeichen aufnehmen. Hat man einem Variablenspeicher eine Zahl zugeordnet, erscheint Fehlermeldung 1, wenn man aus diesem Speicher einen Text abfragt und umgekehrt. Die Fehlermeldung erscheint nur dann nicht, wenn eine Null gespeichert ist. Ein Variablenspeicher kann also nicht gleichzeitig als Zahlen- und Textspeicher verwendet werden.

2.1 Der Basisspeicher

Die 26 Plätze des Basisspeichers sind den Buchstaben A bis Z zugeordnet. Jeder dieser 26 Plätze kann 4 verschiedene Adressen erhalten; der dritte Speicher etwa die Adressen C; A(3); C$ und A$(3). Man achte darauf, daß man in einem Programm nicht verschiedene Namen für den gleichen Speicher verwendet.

Beispiele:					
	C	Basisspeicherplatz	3	(Zahlenvariable)	
	A(25)	"	25	"	
	F$	"	6	(Zeichenvariable)	
	A$(3)	"	3	"	
	A(48 – 25)	"	23	(Zahlenvariable)	
	A(14.7)	"	14	"	
	A(L)	"	L	"	(L ist der Inhalt von Speicher L)

2.2 Der flexible Speicher

Der flexible Speicher kann neben Programmen auch Variablen aufnehmen, die jedoch nur indiziert mit den Adressen A(27) bis A(204) oder A$(27) bis A$(204) eingegeben werden können.
Nimmt man alle Variablenplätze in Anspruch, steht kein Speicherplatz für Programme zur Verfügung.
Wie aus der Abbildung ersichtlich ist, haben die Variablenadressen im Speicher eine feste Position und laufen sequentiell von oben nach unten, während die Programmschritte sequentiell von unten nach oben laufen.
Hat ein Programm z. B. 20 Schritte, so sind drei Variablenplätze A(204), A(203), A(202) belegt und stehen nicht mehr zur Verfügung ($20 \div 8 \approx 3$).
Die höchste zur Verfügung stehende Variablenadresse kann mit dem Kommando MEM [ENTER] ermittelt werden, im o. g. Beispiel ergibt MEM [ENTER]

| 1404 STEPS 175 MEMORIES |

Durch Addition von 175 MEMORIES und 26 Basisspeicher erhält man die Variablenadresse A(201).
Bei Programmänderung durch Einfügen oder Löschen von Zeichen ändert sich die Adresse entsprechend.
Enthält ein Speicherplatz mit höherer Platznummer ordnungsgemäß eine Variable und man schreibt weitere Programme kann es passieren, daß dieser Speicherplatz mit dem Programm überschrieben wird.
Das geschieht selbst dann, wenn ein flexibler Speicherplatz mit niedrigerem Index noch frei ist. Ruft man einen derart überschriebenen flexiblen Speicherplatz ab, erscheint Fehlermeldung 4.

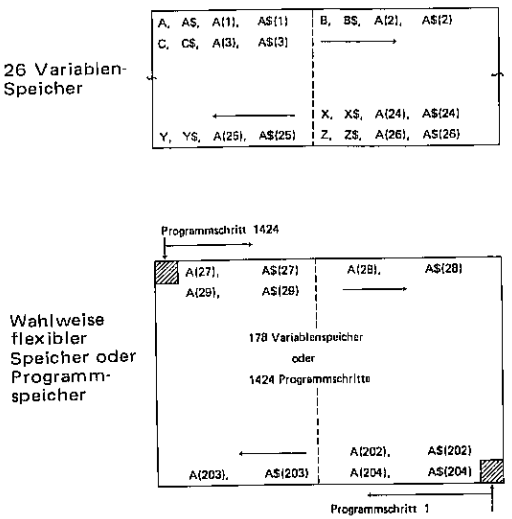
Hinweis

Durch das Kommando MEM wird der flexible Speicher nur hinsichtlich der verbleibenden Programmschrittkapazität abgefragt.
Ist der flexible Speicher nur durch Variable belegt, ergibt das Kommando MEM

| 1424 STEPS 178 MEMORIES |

Die Speicherorganisation zeigt folgendes Schema:

Speicher des PC-1212



3. EINGABE VON VARIABLE

Variablen können manuell eingegeben oder vom Programm abgefragt werden.

3.1 Manuelle Eingabe

Für die manuelle Eingabe gibt es vier Möglichkeiten

(1) Zahlenvariable [=] Ausdruck [ENTER]
Der Ausdruck wird berechnet, rechtsbündig in die Anzeige geschrieben und unter dem gewählten Namen abgespeichert.

Beispiel: C = 2 * 3 [ENTER] 2 * 3 = 6 wird in Speicher C gestellt.

(2) Zahlenvariable 1 [=] Zahlenvariable 2 [ENTER]
Die der Variablen 2 zugeordnete Zahl wird in die Anzeige geschrieben und in den Speicher der Variablen 1 kopiert.

Beispiel: A(4) = C [ENTER] Die 6 im Speicher C wird in Speicher A(4) kopiert.

(3) Zeichenvariable [=] [SHIFT] [↑] Text [SHIFT] [↑] [ENTER]
Der Text wird linksbündig in die Anzeige geschrieben und unter dem gewählten Namen abgespeichert. Überschreitet der Text 7 Zeichen, so werden nur die ersten 7 gespeichert.

Beispiel: B$ = "APFEL" [ENTER] Die Text "APFEL" wird in den Zeichenspeicher B gestellt.

(4) Zeichenvariable 1 [=] Zeichenvariable 2 [ENTER]
Der Text der Zeichenvariablen 2 wird angezeigt und in den Speicher der Variablen 1 kopiert.

Beispiel: A$ = B$ [ENTER] Auch im Speicher A$ ist "APFEL" gespeichert.

3.2 Programmierte Variableneingabe

Die allgemeine Form der Variableneingabe im Programmbetrieb lautet:

[I] [N] [P] [U] [T] Variablenname 1 [SHIFT] [,] Variablenname 2 · · ·

Beispiel: Die Variablen A$ und B sollen im Programmbetrieb abgefragt und angezeigt werden. Als Daten sollen MAERZ und 25 eingegeben werden.

PROGRAMM 4

```
10:  INPUT A$, B
20:  PRINT A$, B
```

Programmeingabe

Eingabe	Anzeige	Anmerkung
"PRO"	>	Betriebsart PRO.
[N] [E] [W] [ENTER]	>	Löschen des Programmspeichers.
10 [I] [N] [P] [U] [T]	10 INPUT _	Eingabeanweisung.
[A] [SHFT] [$] [SHFT] [,]	10 INPUT A$, _	
[B]	10 INPUT A$, B_	
[ENTER]	10 : INPUT A$, B	
20 [P] [R] [I] [N] [T]	20 PRINT _	Ausgabeanweisung.
[A] [SHFT] [$] [SHFT] [,]	20 PRINT A$,_	
[B]	20 PRINT A$, B_	
[ENTER]	20: PRINT A$, B	

Programmabruf

Eingabe	Anzeige	Anmerkung
"RUN"		Betriebsart RUN.
[R] [U] [N] [ENTER]	?	Abfragen von A$.
[M] [A] [E] [R] [Z]	MAERZ _	Eingabe von A$.
[ENTER]	?	Abfragen von B.
25	25 _	Eingabe von B.
[ENTER]	MAERZ 25.	Ausgabe der beiden Variablen.

4. ABRUF VON VARIABLEN

Die Variablen werden in der allgemeinen Form

Variable [ENTER]

abgerufen. Ist der der Variablen zugeordnete Speicher leer, so wird bei Abruf einer Zahlenvariablen 0. angezeigt; bei Abruf einer Zeichenvariablen dagegen wird die gesamte Anzeige gelöscht. Der Ausdruck (–0) wird wie eine Zahl behandelt.

Beispiel: Die Daten des Programms 4 sollen abgerufen werden.

Eingabe	Anzeige	Anmerkung
"RUN"		Betriebsart RUN.
[A] [SHFT] [$]	A $ _	Eingabe des Variablennamens.
[ENTER]	MAERZ	Zeichenvariable A $.
[B]	B _	Eingabe des Variablennamens.
[ENTER]	25	Zahlenvariable B.

5. INDIREKTE ADRESSIERUNG

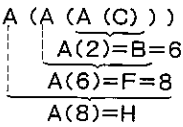
Die indirekte Adressierung einer Variablen gestattet es, den "Namen", die "Adresse" der Variablen in Abhängigkeit von einem Speicherinhalt, der ein Rechenergebnis sein kann, festzulegen. Sie ist nur bei indizierten Variablen möglich.

Beispiele: A(A) Eine Zahlenvariable wird definiert, deren Index gleich dem ganzzahligen Anteil der im Speicher A stehenden Zahl ist.

A$(A(3)) Eine Zeichenvariable wird definiert, deren Index der Zahl in Speicher A(3) = C entspricht.

Die indirekte Adressierung kann durch Verwendung von Klammern bis zu einer Tiefe von 15 Ebenen erweitert werden.

Beispiel: Liest man C = 2, B = 6 und F = 8 ein, so wird durch den Ausdruck A(A(A(C))) die Variable A(8) = H festgelegt, wie man aus untenstehendem Diagramm sieht.



Ist die Adresse einer indirekt adressierten Variablen kleiner als 1 oder größer als es der flexible Speicher unter Berücksichtigung der Programmlänge zuläßt, zeigt der Rechner die Fehlermeldung 4.

Die Vorteile der indirekten Adressierung sieht man an folgenden Beispielen:

Beispiel 1: In einem Programm sollen den Variablen A(2) bis A(26) Zahlen zugeordnet werden. Die direkte Adressierung zeigt die Eingabe entsprechend Programm 5.

PROGRAMM 5

```
10: INPUT A(2), A(3), A(4), ····· A(26)
:
```

Bei diesem Programm muß man umständlich 25 Adressen eingeben. Einfacher geht es mit einem Programm, das den Index beginnend bei 2 (FOR A = 2) bis 26 (TO 26) bei jedem Durchlauf um 1 erhöht. Dies wird automatisch mit der FOR . . . TO . . . NEXT-Anweisung erreicht. Nähere Einzelheiten über diese Anweisung finden sich in Kapitel VIII 3.4.

PROGRAMM 6

```
10 : FOR  A=2TO  26
20 : INPUT  A (A)
30 : NEXT  A
40 : ····
```

Programm-schleife

Beim Programmlauf setzt der Rechner zunächst A = 2, fragt INPUT A(2) ab, lenkt beim NEXT A in einer Schleife zu 10: zurück, setzt A = 3 und so fort. Erst wenn A = 26 erreicht ist, wird die Schleife verlassen und das Programm mit 40 weitergeführt.

Die indirekte Adressierung ist auch bei Zeichenvariablen möglich. Im folgenden Beispiel seien in den Speichern A(1) bis A(26) die Namen von Früchten gespeichert. Durch Eingabe einer Kennummer, z.B. 3, soll der zur Kennummer gehörige Name, im Beispiel "BANANE" aufgerufen werden. Das Programm für diese Aufgabe und seine Eingabe ist unten angegeben.

PROGRAMM 7

```
10 : INPUT  A(27)
20 : PRINT  A$ (A(27))
```

Eingabe

"PRO"
[N] [E] [W] [ENTER]
10 [I] [N] [P] [U] [T] [A] [(] 27 [)] [ENTER]
20 [P] [R] [I] [N] [T]
[A] [SHFT] [$] [(] [A] [(] 27 [)] [)] [ENTER]

Die Eingabe der ersten Texte lautet:

Speicher	Text
A (1)	ORANGE
A (2)	APFEL
A (3)	BANANE
A (4)	MELONE
⋮	⋮

Eingabe

"RUN"
[A] [SHFT] [$] [=] [SHFT] ["]
[O] [R] [A] [N] [G] [E] [SHFT] ["] [ENTER]
[B] [SHFT] [$] [=] [SHFT] ["]
[A] [P] [F] [E] [L] [SHFT] ["] [ENTER]
⋮ ⋮ ⋮

Abruf der codierten Früchtenamen:

Eingabe	Anzeige	Anmerkung
"RUN"		Betriebsart RUN.
[R] [U] [N] [ENTER]	?	Abfrage der Kennummer.
2	2 _	Kennummer 2.
[ENTER]	APFEL	Zugeordneter Früchtename.

VIII PROGRAMMANWEISUNGEN

Anweisungen veranlassen den Rechner, Operationen auszuführen, die dem Fortgang der Rechnung dienen.
Um die Schreibweise übersichtlicher zu gestalten, wird bei der Angabe der allgemeinen Form auf die detaillierte Darstellung mit einzelnen Tastenangaben verzichtet und sofort der Klartext geschrieben. Dabei ist besonders auf die Satzzeichen zu achten. Übersieht man etwa ein Komma zwischen zwei Eingaben, führt das bereits zu einem Programmfehler.

1. EINGABEANWEISUNGEN

Eingabeanweisungen dienen dazu, die Programme mit den nötigen Daten zu versorgen. Mit Hilfe der Wertzuweisung LET legt man diese Daten einmal am Anfang des Programms fest, kann sie dann aber beim Programmlauf nicht mehr ändern. Mit der INPUT-Anweisung dagegen arbeitet man flexibler: Die Anweisung INPUT reserviert lediglich einen Variablennamen. Die Daten für diese Variable werden erst beim Programmlauf abgefragt. Damit ist echter Dialogbetrieb möglich. Die Anweisung AREAD schließlich bietet eine zusätzliche direkte Variableneingabe bei Programmen mit Programmnamen.

1.1 Wertzuweisung (LET)

Die Anweisung LET ordnet einer Variablen einen festen Zahlenwert oder Ausdruck zu. Sie entspricht also der manuellen Variableneingabe. Für jede Variable ist eine eigene Anweisung nötig. Eine Änderung der Zahlenwerte beim Programmlauf ist nicht möglich. Der Variablenname muß immer *links* stehen. Ihm wird der rechts stehende Ausdruck zugeordnet. Der LET-Befehl darf im Allgemeinen weggelassen werden; ausgenommen davon ist LET in Verbindung mit IF (Siehe VIII 3.2).
Für die Wertzuweisung gibt es folgende Formen:

(1) LET Zahlenvariable = Ausdruck

Beispiel:		
	LET A = 5 * 3	15 wird Speicher A zugewiesen.
	LET A = 123	Eingabe von 123 in Speicher A. LET darf hier weggelassen werden.
	A(30) = 3 * 6	18 wird ohne LET Speicher A(30) zugewiesen.
	A(2 * B) = C + D	C + D wird dem Speicher mit dem Index 2B zugewiesen.

(2) LET Zahlenvariable = Zahlenvariable

Beispiel: LET A = B — Die Zahl in B wird in Speicher A kopiert.

(3) LET Zeichenvariable = "Text"

Beispiel: LET Z$ = "BASIC" — Das Wort BASIC wird der Zeichenvariablen Z$ zugewiesen.

Der Text muß in Anführungsstrichen stehen. Überschreitet der Text 7 Zeichen, so werden nur die ersten 7 gespeichert; die überzähligen gehen verloren.

(4) LET Zeichenvariable = Zeichenvariable

Beispiel: C$ = A$ — Der A$ zugeordnete Text wird in C$ kopiert.

Die Programmsätze (1) bis (4) können, getrennt durch ein Komma, in einer Programmzeile aneinandergereiht werden. Dann darf aber LET nur einmal am Anfang stehen.

Beispiel: 10 : LET A = 2, B = 7, C$ = "A = 2; B = 7"
20 : PRINT C$

In Programmzeile 10 wird A die Zahl 2 und B die Zahl 7 zugewiesen. Der Text "A = 2; B = 7" wird in C$ gespeichert und mit Programmzeile 20 angezeigt.

1.2 Eingabe mit der INPUT-Anweisung

Das Wort INPUT teilt dem Rechner mit, daß während des Programmlaufs Daten angefordert werden sollen. Auf INPUT folgt die Liste der Variablen in beliebiger Reihenfolge getrennt durch Kommas. Der Programmlauf wird beim Erreichen einer INPUT-Anweisung unterbrochen und nach der Dateneingabe mit [ENTER] fortgeführt.

1.21 Allgemeine Formen der INPUT-Anweisung

Die Anzweisung INPUT kann in drei verschiedenen Formen verwendet werden:

(1) INPUT Variable, Variable . . . . .

Beispiel: INPUT A, B, C

Verwendet man diese allgemeine Form, so werden die Daten beim Programmlauf der Reihe nach mit einem Fragezeichen in der Anzeige angefordert. Man benötigt also schriftliche Aufzeichnungen über die Reihenfolge der Variablen.

(2) INPUT "Text", Variable, "Text", Variable . . . . .

Beispiel: INPUT "A=", A, "B=", B, "C=", C

Bei dieser allgemeinen Form erscheint bei der Datenanforderung anstelle des Fragezeichens der in den Anführungszeichen formulierte Text. Dieser Text wird im Programmregister nach dem Modus 1 Zeichen gleich 1 Programmschritt gespeichert, sodaß seine Länge nicht auf 7 Zeichen begrenzt ist. Sobald man Daten eingibt, verschwindet der Text, sodaß vor Fortführung des Programmlaufs mit [ENTER] allein die eingegebenen Daten in der Anzeige erscheinen.

(3) INPUT "Text"; Variable, "Text"; Variable . . . . .

Beispiel: INPUT "C="; C, "D="; D

Diese Form unterscheidet sich von (2) nur durch den Strichpunkt zwischen dem Text und der Variablen anstelle des Kommas. Auch bei Form (3) erscheint zur Datenanforderung ein Text. Er bleibt jedoch auch nach der Dateneingabe stehen, sodaß vor Fortführung des Programmlaufs mit [ENTER] Text und Daten in der Anzeige zu sehen sind. Die Form (3) ist also am übersichtlichsten und wird daher meist vorgezogen.
Die drei verschiedenen Formen der programmierten Dateneingabe zeigt Programm 8.

PROGRAMM 8

Programmzeile	Anmerkung
10 : INPUT A, B	Form (1)
20 : INPUT "C=", C, "DICHTE D=", D	Form (2)
30 : INPUT "GESCHW. V IN M/S="; V	Form (3)

Beim Programmlauf in Betriebsart RUN werden die Daten dann wie folgt angefordert:

Eingabe	Anzeige	Anmerkung
"RUN"	>	Betriebsart RUN.
R U N ENTER	?	Programmanforderung; A = ?(1).
3	3_	A = 3
ENTER	?	Anforderung von B (1).
4	4_	B = 4
ENTER	C=	Anforderung von C (2).
5	5_	C = 5
ENTER	DICHTE D=	Anforderung von D (2).
6	6_	D = 6
ENTER	GESCHW. V IN M/S= _	Anforderung von V (3).
7	GESCHW. V IN M/S= 7_	V = 7 m/s
ENTER	>	Ende der Eingabe.

Die Eingabeformen (1), (2) und (3) können gemischt eingesetzt werden.

Beispiel: INPUT A, "B=", B, "NAME?"; C$

Als Variable kann man Zahlen- und Zeichenvariable verwenden. Werden beim Programmlauf Daten für Zeichenvariable angefordert, gibt man sie in der Form "Text" ein.

Gibt man beim Programmlauf zu Korrekturzwecken an Stelle von Daten CL ein, wird der Rest der Programmzeile nicht mehr beachtet. Bei (1) bleibt nach Druck auf CL das Fragezeichen stehen; bei (2) erscheint anstelle des Textes ein Fragezeichen und bei (3) bleibt der Text stehen. Erscheint nach der Dateneingabe wegen irgendeines Programmfehlers die Fehlermeldung, meldet sich der Rechner nach dem Löschen mit CL ebenfalls in der oben angegebenen Form. Bei (2) und (3) dürfen die Variablennamen nur die Form C, A(15) oder A(B) nicht jedoch A(A(30)) oder A(5*9) haben.

1.22 Überspringen einer Zeile

Werden beim Programmlauf Daten angefordert und man drückt statt dessen ENTER, so überspringt der Rechner den Rest der Programmzeile und fährt mit der nächsten Zeile fort. Das sieht man an Programm 9, in dem der Mittelwert der eingegebenen Zahlen berechnet wird. Dieses Programm arbeitet wie folgt: Zunächst löscht man alle Speicher (CLEAR). In Zeile 20 werden die Zahlen angefordert. und zum Speicher B addiert (B = B + A). Gleichzeitig ermittelt man die Anzahl der eingegebenen Werte (C = C + 1). Das Löschen in Zeile 10 war nötig, damit vor der ersten Zahleneingabe 0 in den Speichern B und C steht. Da mehrere Zahlen eingegeben werden sollen, springt das Programm zu Zeile 20 zurück (GOTO 20). Sind alle Zahlen eingegeben, wählt man mit ENTER ohne Zahleneingabe die Zeile 30, überspringt also den GOTO 20-Befehl. Zeile 30 berechnet den Mittelwert (B/C) und Zeile 40 gibt den Befehl zur Anzeige des Mittelwertes.

PROGRAMM 9

Programmzeile	Anmerkung
10 : "A" : CLEAR	Löschen aller Speicher.
20 : INPUT "ZAHL A=" ; A : B=B+A : C=C+1 : GOTO 20	Dateneingabe und Datensumme. Anzahl der Daten und Rücksprung.
30 : D=B/C	Berechnung des Mittelwerts.
40 : PRINT "MITTELWERT="; D	Anzeige von D.
50 : END	Ende des Programms "A".

Beim Programmlauf der, da man dem Programm den Namen "A" gegeben hat, in Betriebsart DEF ausgeführt wird, soll der Mittelwert der Zahlen 12, 24, 19 und 23 berechnet werden.

Eingabe	Anzeige	Anmerkung
"DEF"	>	Betriebsart DEF.
SHIFT A	ZAHL A = _	Anforderung von Zahl 1.
12 ENTER	ZAHL A = _	Zahl 1 und Anforderung von 2.
24 ENTER	ZAHL A = _	Zahl 2 und Anforderung von 3.
19 ENTER	ZAHL A = _	Zahl 3 und Anforderung von 4.
23 ENTER	ZAHL A = _	Zahl 4 (Letzter Wert).
ENTER	MITTELWERT = 19.5	Sprung zu Zeile 30 mit ENTER .
ENTER	>	Programmende.

1.3 Eingabe mit der AREAD-Anweisung

Mit der Eingabeanweisung AREAD (automatic read) weist man einem mit Namen versehenen Programm (Kapitel V 9) automatisch Daten zu. Der Befehl AREAD kann also nur in der Betriebsart DEF Daten anfordern. Er wird überlesen, wenn er in einem nicht mit Namen versehenen Programm auftaucht. Die allgemeine Form der Dateneingabe mit AREAD lautet:

AREAD Variable

Beispiele: AREAD A
AREAD C$

Steht die Eingabeanweisung AREAD Variable direkt im Anschluß an einer PRINT-Anweisung der Form PRINT A, B wird der Variablen der Wert von B zugeordnet; bei der Form PRINT A; B; C; D; · · · · dagegen erhält die Variable den Wert von A.

Beispiel: Zinseszinsrechnung. Wird ein Guthaben P mit I% jährlich verzinst, so ist es in N Jahren auf $F = P * (1 + I/100)^N$ angewachsen. Mit Hilfe der AREAD-Anweisung sollen I, N und P eingelesen werden.

Programmzeile	Anmerkung
10 : "A" : AREAD I	Anforderung von I.
20 : I = I / 100	I%
30 : END	Ende des Programms "A".
40 : "S" : AREAD N	Anforderung von N.
50 : END	Ende des Programms "S".
60 : "D" : AREAD P	Anforderung von P.
70 : END	Ende des Programms "D".
80 : "F" : F = P * (1 + I) ^ N	Berechnung von F.
90 : PRINT F	
100 : END	Ende von Programm "F".

Der Programmlauf muß in der Betriebsart DEF stattfinden.

Eingabe	Anzeige	Anmerkung
"DEF"	)	Betriebsart DEF.
6.8 SHIFT A	)	I = 6.8% Zinsen
4 SHIFT S	)	N = 4 Jahre
5000 SHIFT D	)	P = 5000 DM
SHIFT F	6505.115547	F = 6505.16 DM
5 SHIFT S	)	Neue Laufzeit 5 Jahre.
SHIFT F	6947.463404	F = 6947.46 DM

2. AUSGABEANWEISUNGEN

Ausgabeanweisungen dienen dazu, Daten programmgesteuert in der gewünschten Form auszugeben. Die Anweisung PRINT veranlaßt Datenausgabe mit maximaler Stellenzahl und Stop der Programmausführung. Bei PAUSE wird ebenfalls die maximale Stellenzahl angezeigt; das Programm läuft jedoch nach kurzer Zeit automatisch weiter. Mit USING, PRINT USING und PAUSE USING schließlich kann man das Format der ausgegebenen Daten frei wählen.

Textausgaben können entsprechend der Kapazität des Eingaberegisters 80 Zeichen umfassen; angezeigt werden jedoch nur die ersten 24 Zeichen.

2.1 Die PRINT-Anweisung

Die PRINT-Anweisung veranlaßt die Anzeige der Daten mit der vollen Stellenzahl. Die Werte der Zahlenvariablen können maximal 16 Stellen der Anzeige belegen; bei Zeichenvariablen werden maximal 7 Stellen benötigt. Der Programmlauf wird nach der Datenanzeige unterbrochen, bis man ihn mit ENTER wieder startet.

Die Anweisung PRINT kann in fünf verschiedenen Formen verwendet werden:

(1) PRINT Ausdruck

Beispiele: PRINT 123 + 456
PRINT A

Diese Form der Ausgabeanweisung veranlaßt die Anzeige des Rechenergebnisses oder Speicherinhaltes. Die Anzeige erfolgt rechtsbündig mit maximal 16 Stellen.

(2) PRINT "Text"

Beispiel: PRINT "DICHTE="

Diese Ausgabeform veranlaßt die Anzeige des zwischen den Anführungszeichen stehenden Textes. Der Text wird linksbündig geschrieben.

(3) PRINT Zeichenvariable

Beispiel: PRINT A$

Die Ausgabeanweisung (3) schreibt den der gewünschten Zeichenvariablen zugeordneten Text, also bis zu 7 Zeichen, linksbündig in die Anzeige.

Die Verwendung der Eingabeformen (1) bis (3) zeigt Programm 11

PROGRAMM 11

Programmzeile	Anmerkung
10 : INPUT A$	Eingabeanweisung.
20 : PRINT 5*6	Ausgabe in Form (1).
30 : PRINT "PROGRAMM A"	Ausgabe in Form (2).
40 : PRINT A$	Form (3); der in Zeile 10 eingegebene Text wird angezeigt.

Dieses Programm läuft in Betriebsart RUN wie folgt:

Eingabe	Anzeige	Anmerkung
"RUN"		Betriebsart RUN.
R U N ENTER	?	Start und Abfrage von A$.
W E L T ENTER	30.	A$ = WELT; Form (1).
ENTER	PROGRAMM A	Form (2).
ENTER	WELT	Form (3).

(4) PRINT { Ausdruck / "Text" / Zeichenvariable } , { Ausdruck / "Text" / Zeichenvariable }

Beispiel: PRINT A, B
PRINT "A=", A

Bei dieser Ausgabe von *zwei* Blöcken wird die Anzeige in zwei zwölfstellige Bereiche unterteilt. Der linke Block steht im linken Anzeigenbereich; der rechte im rechten Bereich. Zeichen werden linksbündig, Ziffern rechtsbündig geschrieben. Übersteigt eine Zahl 12 Stellen, werden die letzten Stellen der Mantisse weggelassen. Bei langen Texten werden nur die ersten 12 Zeichen geschrieben.

Beispiel: Programm 12 wandelt Polarkoordinaten in rechtwinklige Koordinaten um. Eingegeben werden R = r und C = φ; angezeigt werden X = x und Y = y.

PROGRAMM 12

```
10 : INPUT R, C
20 : X=R*COS C : Y=R*SIN C
30 : PRINT X, Y
40 : END
```

Im Programmlauf sollen die rechtwinkligen Koordinaten des Zeigers (r = 12; φ = 30°) bestimmt werden.

Eingabe	Anzeige	Anmerkung
"RUN" "DEG"		Betriebsart RUN; Winkeleinheit Grad (DEG).
R U N ENTER	?	
12 ENTER	?	r = 12
30 ENTER	10. 39230485 (x-Koordinate) 6. (y-Koordinate)	φ = 30

(5) PRINT { Ausdruck / "Text" / Zeichenvariable } ; { "Text" / Zahlenvariable / Zeichenvariable } ; { "Text" / Zahlenvariable / Zeichenvariable } ; .....

Beispiele: PRINT "A="; A; "B="; B; "C="; C
PRINT A$; B C$; D

Möchte man mehr als zwei Daten gleichzeitig anzeigen, verwendet man die Form (5). Dazu sind die Blöcke anstelle des Kommas mit Strichpunkten zu trennen. Um eine übersichtliche Darstellung zu erreichen, tastet man bei Texten als erstes ein Leerzeichen SPC ein. Zuviele Ausgaben in einer Zeile sind nicht sinnvoll, da immer nur die ersten 24 Stellen zu sehen sind und somit die Gefahr besteht, daß der rechte Teil der Anzeige abgeschnitten wird.

Der zweite und jeder weitere Block darf nur einen Text, eine Zahlen- oder Zeichenvariable enthalten, nicht jedoch eine Zahl oder einen mathematischen Ausdruck. Die Variablen können in der Form C, A(30), A$(C) nicht jedoch in der Form A(A(30)) oder A(2 * 3) geschrieben werden.

Programm 13 zeigt die Anwendung der Form (5). Es handelt sich um ein Programm ähnlich Programm 9; angezeigt werden Datensumme und Datenanzahl.

PROGRAMM 13

Programmzeile	Anmerkung
10 : "A" : CLEAR	Löschen der Speicher.
20 : INPUT "SUMMAND="; A	Zahlenanforderung.
30 : B=B+A : C=C+1	Summe und Datenanzahl.
40 : PRINT "SUMME="; B; "ANZAHL="; C	Anzeige in Form (5).
50 : GOTO 20	

Mit diesem Programm soll die Summe der Zahlen 456 und 789 ermittelt werden.

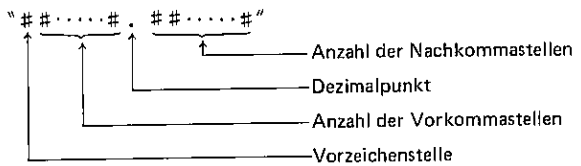
Eingabe	Anzeige	Anmerkung
"DEF"	>	Betriebsart DEF.
[SHIFT] [A]	SUMMAND= _	A1 = 456
456 [ENTER]	SUMME= 456. ANZAHL = 1.	Ausgabeform (5).
[ENTER]	SUMMAND = _	A2 = 789
789 [ENTER]	SUMME= 1245. ANZAHL = 2.	

2.2 Die PAUSE-Anweisung

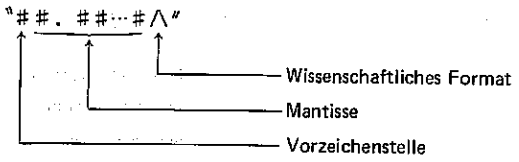
Die Anweisung PAUSE wird genauso wie PRINT verwendet. In den allgemeinen Formen ist einfach das Wort PRINT durch PAUSE zu ersetzen. Unterschiedlich ist lediglich der Programmlauf. Während bei PRINT der Programmlauf nach der Anzeige der Daten auf Dauer gestoppt ist und manuell mit [ENTER] neu gestartet werden muß, läuft er bei PAUSE automatisch etwa 0,85 Sekunden nach Anzeige der Daten wieder an.

2.3 Die USING-Anweisung

Die Anweisung USING bestimmt das Format, in dem Zahlen im Anschluß an PRINT- oder PAUSE-Anweisungen angezeigt werden.
Zur Festlegung des Formats dienen die Zeichen [#] , [.] und [^] . [#] bestimmt die Anzahl der Stellen; [.] legt den Dezimalpunkt fest und [^] führt zur Anzeige im wissenschaftlichen Format. Gibt man nur eine Folge der Zeichen [#] ein, so wird lediglich der ganzzahlige Anteil einer Zahl angezeigt. Wegen eines möglichen Vorzeichens muß stets ein [#] mehr gesetzt werden, als die anzuzeigende Zahl Vorkommastellen besitzt; eine dreistellige Zahl benötigt damit das Format # # # # . Die Anzahl der Zeichen [#] nach dem Dezimalpunkt legt die Anzahl der Nachkommastellen fest.
Das allgemeine Format lautet damit



oder



Gibt man beim wissenschaftlichen Format mehr als zwei Vorkommastellen ein, wird das nicht beachtet. Auch die Verwendung mehrerer [^] hat keinen Einfluß auf die Anzeige. Kann eine Zahl nicht im geforderten Format dargestellt werden, wird Fehler mit Code 6 gemeldet.

Folgende Formen der USING-Anweisung sind möglich:

(1a) USING "Format"
(1b) USING

Die Formatvereinbarung (1a) gibt die Anweisung, daß *alle* folgenden Zahlen im spezifizierten Format angezeigt werden. Mit der Anweisung (1b), also USING ohne Formatangabe löscht man die vorausgegangene Formatanweisung. Die auf (1b) folgenden Zahlen werden so, wie beim manuellen Rechnen, angezeigt.

Die Verwendung der Formatanweisung (1) sieht man an Programm 14. Es soll die Zahl 123.4567891 in verschiedenen Formaten dargestellt werden.

PROGRAMM 14

```
10 : A = 123.4567891
20 : USING "Format"
30 : PRINT A
```

Die Anzeige bei den unterschiedlichsten Formaten sieht man aus der folgenden Tabelle:

Format in Zeile 20	Angezeigte Zahl
Format (1b)	123. 4567891
#	Fehlermeldung mit Fehlercode 6
##	
###	
####	123
#####	123
####.	123.
######. #	123. 4
#######. ###	123. 456
####. ########	123. 4567891
####. ##########	123. 456789100
##. ###^	1. 234 E 02
####. ######^^^	1. 234567 E 02
###. ##########^	1. 234567891 E 02
. ##########^	1. 234567891 E 02

(2a) PRINT / PAUSE USING "Format"; ·····

(2b) PRINT / PAUSE USING; ·····

Beispiel: PRINT USING "###.##"; A

Die Formatanweisung (2a) ermöglicht es, daß jede Anweisung in einem anderen Format angezeigt wird. Nach der Formatanweisung muß ein Strichpunkt stehen. Mit (2b) wird die Formatanweisung gelöscht. Programm 15 demonstriert diese Formatanweisung.

PROGRAMM 15

```
10 : A=-123.456
20 : PAUSE USING "####" ; A
30 : PAUSE USING "####. #" ; "A=" , A
40 : PAUSE "A=" , USING "####. #" ; A
50 : PAUSE A , USING "####. ##" ; A
60 : PAUSE A ; USING "####" ; A
70 : PAUSE USING ; A
```

Die verschiedenen Möglichkeiten der Anzeige der Zahl −123.456 zeigt der Programmlauf:

Eingabe	Anzeige
"RUN"	
R U N ENTER	−123
	A= −123. 4
	A= −123. 4
	−123. 45 −123. 45
	−123. 45−123
	−123. 456

Werden, wie das in Zeile 50 der Fall ist, zwei Ausgaben entsprechend VIII 2.1 (4) durch ein Komma getrennt, so wird das Format für die *ganze* Programmzeile durch die Formatanweisung *nach* dem Komma bestimmt. Dies gilt auch dann, wenn man vorher für den ersten Teil ein anderes Format festgelegt hat. Trennt man dagegen die Ausgaben mit einem Strichpunkt gemäß VIII 2.1 (5), so sind für beide Zahlen verschiedene Formate möglich, wie das Zeile 60 zeigt.

3. STEUERANWEISUNGEN

In einem BASIC-Programm werden die Anweisungen in aufsteigender Zeilennummer durchgeführt. Damit steht die Bearbeitungsfolge schon vor dem Programmlauf fest. Möchte man den Programmlauf jedoch in Abhängigkeit von eingegebenen oder berechneten Werten steuern, so stehen dafür eine Reihe spezieller Steueranweisungen zur Verfügung.

3.1 Unbedingte Sprunganweisung GOTO

Die Anweisung GOTO steuert den Programmlauf ohne jede einschränkende Bedingung zu einer bestimmten Zeile, dem Sprungziel. Als Sprungziel kann die Zeilenzahl oder eine Marke (Label) dienen. Das GOTO kann hinter einem Programmsatz stehen. *Es darf jedoch auf keinen Fall ein weiterer Programmsatz auf GOTO folgen.* GOTO mit Sprungziel muß also ausnahmslos ans Zeilenende gesetzt werden.

Folgende allgemeine Formen können mit GOTO gebildet werden:

(1) GOTO Ausdruck

```
          GOTO 10
Beispiele: GOTO 5*9
          GOTO A
```

Die Sprunganweisung (1) lenkt den Programmlauf zu der Zeile, deren Nummer dem gazzahligen Anteil des Ausdrucks entspricht. Es sind Sprungziele zwischen 1 und 999 möglich; andere Werte führen zur Fehlermeldung 2. Das Sprungziel kann direkt und indirekt eingegeben werden.

Direkte Eingabe des Sprungziels:

```
20 : INPUT "ZAHL A="; D
80 : GOTO 20
```

PROGRAMM 16

Indirekte Eingabe des Sprungziels

```
10 : LET A = B + C
20 : GOTO A
30 : PRINT "E1"   <- A = 30
40 : PRINT "E2"   <- A = 40
50 : PRINT "E3"   <- A = 50.1
```

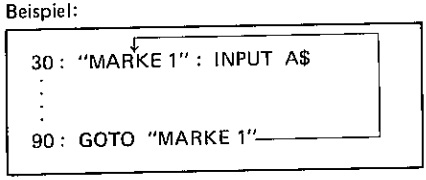
PROGRAMM 17
Das Sprungziel hängt von der Größe des berechneten Wertes A ab.

(2) GOTO "Text"

Beispiel: GOTO "AB"

Die Sprunganweisung GOTO "Text" lenkt den Programmlauf zu der Zeile, die durch den betreffenden Text mit einer Programmarke versehen ist. Maßgebend für die Sprungadresse sind die ersten 7 Zeichen des Markentextes. Hat der Text mehr Buchstaben, so werden diese nicht beachtet. Der Doppelpunkt nach der Marke darf entfallen.

Beispiel:



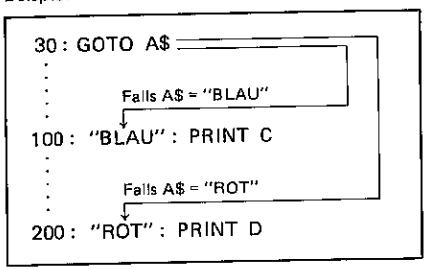
PROGRAMM 18

(3) GOTO Zeichenvariable

Beispiel: GOTO A$

Diese Sprungadresse gibt das Sprungziel indirekt an. Es steht nämlich in dem der Zeichenvariablen zugeordneten Speicher.

Beispiel:



PROGRAMM 19
Ist in A$ der Text "BLAU" gespeichert, springt das Programm zu Zeile 100; lautet der Text "ROT", erfolgt der Sprung nach Zeile 200.

3.2 Programmverzweigungsanweisung IF

Mit der Anweisung IF hat man die Möglichkeit, das Programm abhängig von bestimmten Bedingungen zu verzweigen. Als Bedingung verwendet man den logischen Vergleich zweier Ausdrücke. Die Vergleichsoperatoren sind in IV 6 beschrieben. Allen Verzweigungsformen ist gemeinsam, daß die Programmzeile in der das IF steht dann, wenn die Bedingung erfüllt ist (1 = "wahr"), bis zum Ende ausgeführt wird. Ist die Bedingung dagegen nicht erfüllt (0 = "falsch"), wird sofort, nachdem der Rechner das festgestellt hat, der Rest der Zeile übersprungen und das Programm läuft in der nächstfolgenden Zeile weiter.

Folgende Verzweigungsformen sind vorgesehen:

(1) IF Ausdruck 1 Logischer Vergleich Ausdruck 2 Befehl

Der Befehl kann eine Sprunganweisung, eine Ausgabeanweisung oder eine Wertzuweisung sein. Enthält der Befehl keine Sprunganweisung, bleibt das Programm dann, wenn die Bedingung erfüllt ist, in der betreffenden Zeile stehen. Diese Form wirkt dann wie die Steueranweisung STOP. Schließt sich direkt an einen logischen Vergleich mit IF eine Wertzuweisung an, muß diese ausnahmsweise unbedingt mit LET begonnen werden.

Beispiel: IF B > C LET B = B + 1

Folgt auf den logischen Vergleich mit IF ein Sprungbefehl GOTO, kann stattdessen wie in BASIC üblich, auch THEN geschrieben werden.

Beispiel: IF B >= C THEN 50 ⟷ IF B >= C GOTO 50

Die Arbeit mit der Verzweigungsform (1) zeigt das Programm 20:

```
   ⋮
40 : I F  A*B>=C  PAUSE  A*B : GOTO  90
50 : A=A+1
   ⋮
90 : A=A+B
   ⋮
```

PROGRAMM 20

Ist $A * B \geqq C$ ("wahr"), wird kurzzeitig der Zahlenwert von $A * B$ angezeigt und das Programm verzweigt nach Zeile 90.
Ist $A * B < C$ ("falsch"), fährt das Programm mit der nächstfolgenden Zeile, also 50, fort. Der Rest der Zeile 40, also PAUSE A*B: GOTO 90, wird übersprungen.

(2) IF Ausdruck Befehl

Bei dieser Form der IF-Anweisung wird dann, wenn der Ausdruck größer Null ist, der Befehl ausgeführt. Ist der Ausdruck jedoch kleiner oder gleich Null, wird der Befehl übersprungen und das Programm mit der nachfolgenden Zeile fortgesetzt. Als Beispiel dient Programm 21:

```
   ⋮
30 : I F  AGOTO  80
40 : A=B*C
   ⋮
```

PROGRAMM 21

Ist $A > 0$, verzweigt das Programm nach Zeile 80.
Ist $A \leqq 0$, geht der Programmlauf mit Zeile 40 weiter.

(3) IF { **"Text 1"** / **Zeichenvariable 1** } = { **"Text 2"** / **Zeichenvariable 2** } **Befehl**

Beispiele: IF A$ = "ABC"
IF A$ = B$

Sind die in Anführungszeichen geschriebenen oder den Zeichenvariablen zugeordneten Texte in den Blöcken 1 und 2 gleich, wird der Befehl ausgeführt. Wenn nicht, wird der Befehl übersprungen und das Programm läuft mit der nächstfolgenden Zeile weiter. Ist der Text länger als 7 Zeichen, werden nur die ersten 7 beachtet. Die Verwendung der Form (3) zeigt Programm 22

```
   ⋮
30 : I F  A$= "GUARD" GOTO  100
40 : INPUT  A$
   ⋮
```

PROGRAMM 22

Steht im Speicher A$ das englische Wort GUARD, verzweigt das Programm zu Zeile 100. Steht dort irgendein anderes oder gar kein Wort, fährt das Programm in Zeile 40 weiter.

(4) IF Zeichenvariable Befehl

Ist der Zeichenvariablen irgendein Text zugeordnet, wird der Befehl ausgeführt. Ist der Speicher der Zeichenvariablen dagegen leer, wird der Befehl übersprungen und das Programm mit der nächstfolgenden Zeile fortgesetzt.

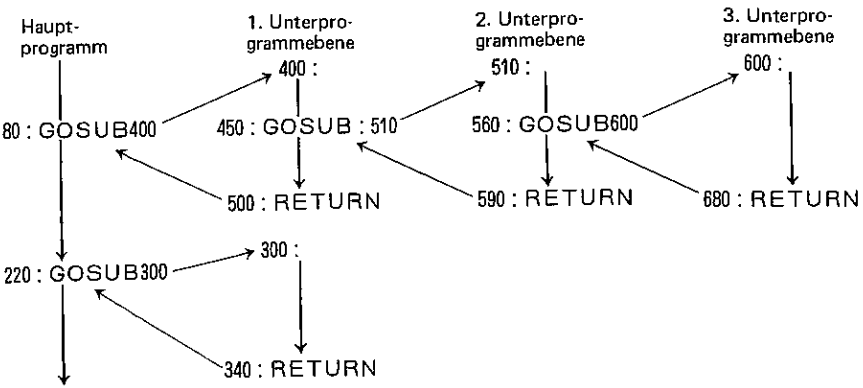
3.3 Unterprogrammtechnik GOSUB RETURN

Soll der gleiche Programmabschnitt mehrmals innerhalb eines Programms verwendet werden, kann man ihn als Unterprogramm schreiben. Dieses Unterprogramm kann dann von einer beliebigen Zeile des Hauptprogramms aus mit GOSUB Unterprogrammadresse aufgerufen werden. Die letzte Zeile des Unterprogramms endet mit RETURN. Das weist den Rechner an, den Programmlauf wieder an die mit GOSUB markierte Ausgangsstelle zurückzulenken. GOSUB wird also eingesetzt wie GOTO. Der Unterschied zum absoluten Sprungbefehl liegt allein im anschließenden Rücksprung zum Ausgangspunkt. *Die RETURN-Anweisung muß als letzter Befehl in einer Programmzeile stehen*; es dürfen keine weiteren Anweisungen auf RETURN folgen. Schreibt man in einem Programm einen RETURN-Befehl ohne vorausgehendes GOSUB, meldet der Rechner diesen Fehler im Satzbau mit Fehlercode 3.

Die allgemeine Form eines Unterprogramms lautet:

GOSUB Unterprogrammadresse Unterprogramm RETURN

Als Unterprogrammadresse kann ein Ausdruck stehen (GOSUB 10, GOSUB A, GOSUB 2 ∗3), ein Text (GOSUB "U.PR.1") oder eine Zeichenvariable (GOSUB A$).
Innerhalb eines Unterprogramms können weitere Unterprogramme aufgerufen werden. Insgesamt sind 4 Unterprogrammebenen möglich. Die Schachtelung von Unterprogrammen zeigt das folgende Diagramm:



Den Einsatz der Unterprogrammtechnik bei der numerischen Integration nach Simpson zeigt das Programm 23. Um das bestimmte Integral $\int_{x_0}^{x_{2p}} y\,dx$ numerisch zu ermitteln, teilt man das Intervall zwischen $x = x_0$ und $x = x_{2p}$ in 2p gleiche Abschnitte – 2p damit es eine gerade Anzahl wird – und verwendet die Formel von Simpson

$$C = \int_{x_0}^{x_{2p}} f(x)\,dx = \frac{h}{3}\,[(y_0 + y_{2p}) + 4(y_1 + y_3 + \cdots\cdots + y_{2p-1}) + 2(y_2 + y_4 + \cdots\cdots + y_{2p-2})]$$

mit $\quad h = \dfrac{(x_{2p} - x_0)}{2p}$

In dieser Formel muß $y(x)$ an mehreren Stellen berechnet werden; man schreibt $y(x)$ daher als Unterprogramm. Wählt man als Beispiel $y = x^3 - 2x^2 - x + 2 = ((x-2)x - 1)x + 2$ so sieht das Unterprogramm in Zeile 500 wie folgt aus:

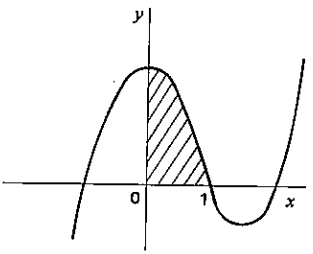
```
500 Y = ((x - 2) * x - 1) * x + 2
510 RETURN
```

Das Programm der Simpson-Regel lautet dann

PROGRAMM 23

```
 10: "A" : INPUT "X0= "; D, "X2P= "; E, "P= "; F
 20: B = (E - D) / 2 / F
 30: A = 0: X = D: GOSUB 500
 40: A = Y + A: X = X + B: GOSUB 500
 50: A = Y * 4 + A: X = X + B: GOSUB 500
 60: A = Y + A: F = F - 1
 70: IF F <> 0 GOTO 40
 80: C = A * B / 3
 90: BEEP 3: PRINT "INT.=", C
100: END
500: Y = ((X - 2) * X - 1) * X + 2
510: RETURN
```

Wählt man $x_0 = 0$ und $x_{2p} = 1$, so wird exakt $\int_0^1 y\,dx = \frac{x^4}{4} - \frac{2x^3}{3} - \frac{x^2}{2} + 2\Big|_0^1 = \frac{13}{12} =$ 1.083333333. Die Näherung nach Simpson liefert bei genügend kleiner Schrittweite das gleiche Ergebnis, wie man beim Lauf des Programms 23 mit $x_0 = 0$, $x_{2p} = 1$ und p = 20 sieht:



Eingabe	Anzeige		Anmerkung
"DEF"			Betriebsart DEF.
SHIFT A	X0 =		
0 ENTER	X2P =		x_0
1 ENTER	P =		x_{2p}
20 ENTER	INT. =	1.083333333	Gesuchtes Integral.

Da das Unterprogramm wegen der großen Anzahl von Schritten sehr oft durchlaufen werden muß, benötigt der Rechner anschließend an die Eingabe von p etwa 40 s. Während dieser Zeit erscheint allein das Symbol RUN in der Anzeige. Ist die Rechnung beendet, meldet sich der Rechner mit 3 akustischen Singalen.

3.4 Schleifenanweisungen FOR TO STEP NEXT

Mit den Schleifenanweisungen erreicht man, daß eine Folge von Programmsätzen mehrfach mit unterschiedlichen Variablenwerten durchlaufen wird. Zwar lassen sich solche Abläufe bereits mit GOTO und IF programmieren; wesentlich geringer ist der Aufwand aber unter Verwendung der Anweisungen FOR, TO, STEP und NEXT.

Möchte man etwa alle Sinuswerte zwischen 0° und 90° in Schritten von 2° berechnen, so ist das "Schleifenprogramm", nämlich die Berechnung des Sinus, immer das gleiche. Geändert wird lediglich das Argument. Der Rechner löst dieses Problem elegant: Mit FOR bestimmt man den Anfangswert, also 0; mit TO den Endwert, also 90. Die Schrittweite 2 wird mit STEP festgelegt. Auf diese Anweisungen folgt das Schleifenprogramm, Berechnung und Anzeige des Sinuswertes. Anschließend weist man den Rechner mit der Anweisung NEXT an, ein neues Argument zu wählen.

Die allgemeine Form der Schleifenanweisung lautet also:

FOR Zahlenvariable = Ausdruck 1 TO Ausdruck 2 STEP Ausdruck 3
Schleifenprogramm
NEXT Zahlenvariable

Das erwähnte Beispiel wird damit wie folgt programmiert:

PROGRAMM 24

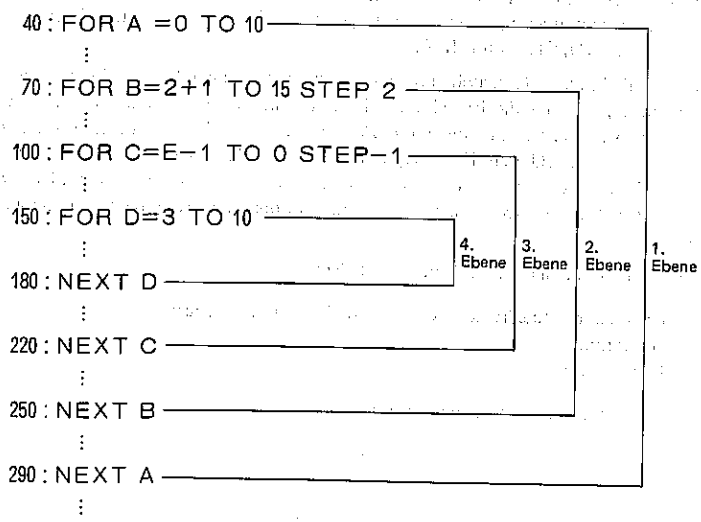
Programmzeile	Anmerkung
10 : FOR A = 0 TO 90 STEP 2	Zahlenvariable A; Anfangswert A = 0 Endwert A = 90; Schrittweite $\Delta A = 2$.
20 : B = SIN A: PRINT B	Schleifenprogramm.
30 : NEXT A	A = A + Δ A ; weiter bei 20:
40 : PRINT "SCHLEIFENENDE"	Falls A = 90.
50 : END	

Nach FOR und NEXT muß immer die gleiche Variable stehen. Das mehrfach durchlaufene Schleifenprogramm steht zwischen FOR und NEXT. Beim ersten Durchlauf des Schleifenprogramms wird der Variablen der Zahlenwert von Ausdruck 1 zugewiesen. Erreicht der Programmlauf die Anweisung NEXT, wird dieser Zahlenwert um das in Ausdruck 3 festgelegte sogenannte Inkrement erhöht oder vermindert. Anschließend wird erneut mit dem geänderten Wert von Ausdruck 1 das Schleifenprogramm durchlaufen. Ist die Schrittweite positiv, muß der Wert des Ausdrucks 2 größer sein als der von Ausdruck 1. Das Programm wird mit der auf NEXT folgenden Zeile fortgeführt, sobald der Zahlenwert der Variablen den Ausdruck 2 erreicht oder überschritten hat. Bei negativem Inkrement muß der Zahlenwert des Ausdrucks 2 kleiner sein als der von Ausdruck 1. Der Schleifendruchlauf wird beendet, wenn der Wert der Variablen den Wert des Ausdrucks 2 erreicht oder unterschritten hat. Läßt man in der allgemeinen Form den Befehl STEP Ausdruck 3 weg, gibt also keine Schrittweite vor, wählt der Rechner automatisch die Schrittweite 1.
Von den Ausdrücken 2 und 3 wird im Programm lediglich der ganzzahlige Anteil verwendet; der Zahlenwert dieser Ausdrücke muß zwischen +1000 und −1000 liegen. Ist der ganzzahlige Anteil von Ausdruck 3 gleich Null oder überschreitet der Wert der Ausdrücke 2 und 3 die angegebenen Grenzen, erscheint Fehlercode 1. Mit Fehlercode 3 wird gemeldet, wenn man bei

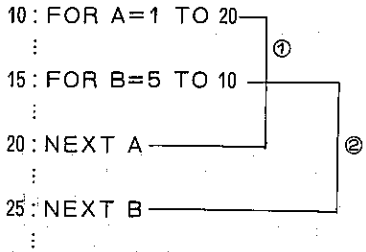
FOR und NEXT verschiedene Variable gewählt hat.
Nimmt man als Variable für die FOR-NEXT-Anweisung einen der Speicher A(1) bis A(22), so benötigt der Schleifendurchlauf mehr Zeit, als wenn man Speicher A(23) und die folgenden verwendet.

FOR-NEXT-Anweisungen können in 4 Ebenen geschachtelt werden.

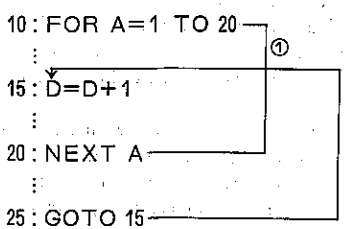
Beispiel:


40 : FOR A =0 TO 10
70 : FOR B=2+1 TO 15 STEP 2
100 : FOR C=E−1 TO 0 STEP−1
150 : FOR D=3 TO 10
180 : NEXT D
220 : NEXT C
250 : NEXT B
290 : NEXT A
4. Ebene
3. Ebene
2. Ebene
1. Ebene

Beim Schachteln dürfen sich die Schleifen aber nicht kreuzen, sonst erscheint Fehlermeldung 3 oder 4, wie man im folgenden Beispiel sieht. Der Rechner erkennt den Fehler in Zeile 25.



Fehler 3 oder 4 erscheint auch, wenn man wie beim folgenden Beispiel einen Sprung mit GOTO ins Schleifenprogramm festlegt. Zwar wird hier die erste Schleife normal abgearbeitet, sobald aber in Zeile 25 der Sprung ins Schleifenprogramm erkannt wird, meldet das der Rechner.

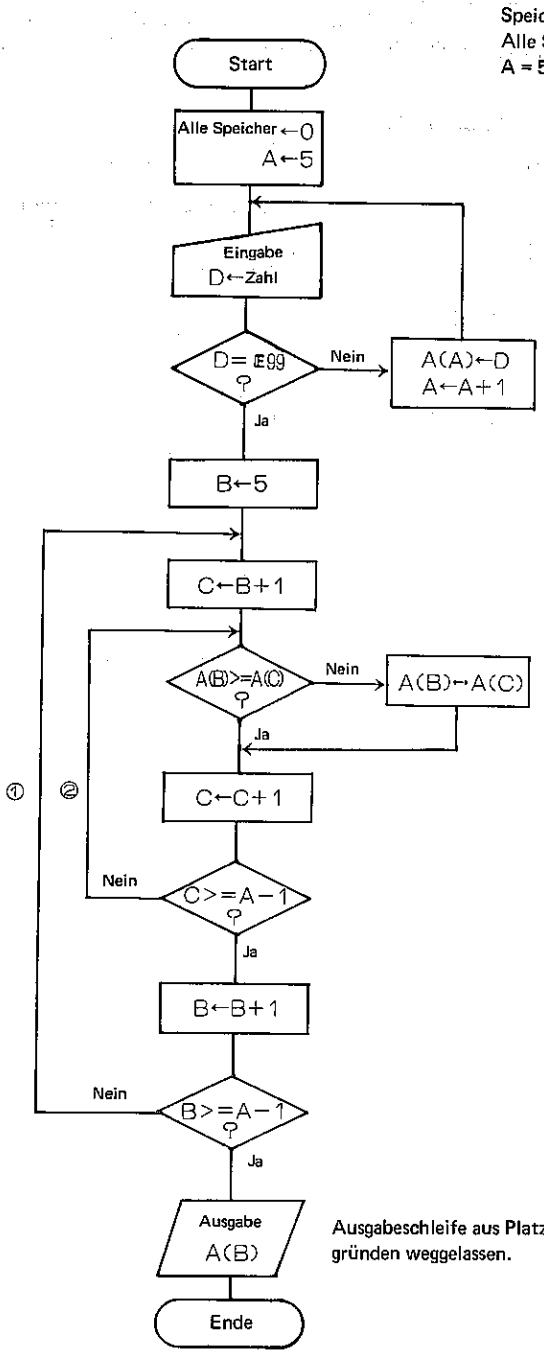


In Programm 25 sind mehrere Schleifen verwendet. Es ist ein Sortierprogramm, das in willkürlicher Reihenfolge eingegebene Zahlen nach fallenden Werten ordnet. Die Zahlen müssen vor ihrer Sortierung gespeichert werden; damit ist ihre Anzahl natürlich begrenzt. Die Eingabe E99 dient dazu, dem Rechner mitzuteilen, daß alle Zahlen eingegeben sind und er mit dem Sortieren beginnen soll.

PROGRAMM 25

Programmzeile	Anmerkung
10: "A" CLEAR :A=5	Programmname "A"; Speichervorbereitung.
20: INPUT " ZAHL =" ; D	Eingabeprogramm.
30: IF D=E99GOTO 50	
40: A (A) =D : A=A+1 : GOTO 20	
50: FOR B=5TO A-1	② ① Hauptprogramm mit zwei Schleifen.
60: FOR C=B+1TO A-1	
70: IF A (B) > =A (C) GOTO 110	
80: D=A (B)	
90: A (B) =A (C)	
100: A (C) =D	
110: NEXT C	
120: NEXT B	
130: "B" FOR B=5TO A-1	③ Ausgabeprogramm "B" mit dritter Schleife.
140: BEEP 2 : PRINT B-4 , A (B)	
150: NEXT B	
160: BEEP 5 : END	

Das folgende Flußdiagramm erläutert den Programmlauf:



Speichervorbereitung:
Alle Speicher löschen;
A = 5

Eingabeschleife. Sie wird durch Eingabe von E 99 verlassen.

Sortierung

Ausgabeschleife aus Platzgründen weggelassen.

Hat man in Stellung RUN alle Zahlen eingelesen, ruft man das Sortierprogramm mit E99 ab. Nach Ertönen des Signals erscheint die Ausgabe in der Form Ordnungsnummer links/ Zahl rechts. Die Funktion des Sortierprogramms erkennt man am besten beim schrittweisen Ausführen mit dem Befehl DEBUG.

3.5 Programmunterbrechung STOP

Das Programm kann an einer beliebigen Stelle mit der programmierten Steueranweisung STOP unterbrochen werden. Man wendet STOP bevorzugt an einem logischen Ende eines mehrfach verzweigten Programms an. Sobald der Programmlauf den Befehl STOP erreicht hat, meldet sich der Rechner mit BREAK AT Zeilennummer. Nun sind manuelle Rechnungen möglich. Soll im Programm weitergefahren werden, gibt man den Befehl CONT ein.

Beispiel:

PROGRAMM 26

```
100 : C=3*7
200 : STOP
300 : PRINT "C=" : C
    :
```

Beim
Der Programmlauf wird in Zeile 200 wie unten gezeigt unterbrochen.

Eingabe	Anzeige	Anmerkung
"RUN" [R][U][N][ENTER]	>BREAK AT 200	Betriebsart RUN. Unterbrechungsanzeige
[C]	C_	
[ENTER]	21.	Manuelles Rechnen ist möglich.
4 [*] 8 [ENTER]	32.	
[C][O][N][T]	CONT_	Fortsetzung des Programms mit dem Befehl CONT.
[ENTER]	C=21.	

3.6 Programmende END

Am Ende jedes Programms muß die Steueranweisung END stehen. Sie teilt dem Rechner mit, daß das Programm beendet ist, und ein neues beginnt. Ist nur ein Programm gespeichert, kann man auf END verzichten.

3.7 Löschanweisung CLEAR

Trifft der Programmlauf auf ein programmiertes CLEAR, werden alle Datenspeicher, also Basisspeicher und flexible Speicher, gelöscht. Die Programme und die im RESERVE-Speicher stehenden Ausdrücke bleiben jedoch weiter gespeichert. Einzelne Speicher löscht man mit der Routine

Speichername = 0

3.8 Winkeleinheit DEGREE RADIAN GRAD

Auch die bereits bei den Trigonometrischen Funktionen besprochenen Winkeleinheiten können programmiert werden. Der Name der Winkeleinheit wird dabei einfach als Programmsatz geschrieben.

Beispiel: Man berechne den Sinus von A in Grad und Radiant.

PROGRAMM 26

```
10 : INPUT A
20 : DEGREE: B = SIN A: PRINT B
30 : RADIAN: B = SIN A: PRINT B
40 : END
```

3.9 Akustisches Signal BEEP

Mit dem Befehl BEEP programmiert man kurze Piepstöne. Der Befehl ist vorallem bei Programmen mit langer Laufzeit interessant. Die allgemeine Form lautet:

Beep Ausdruck

Beispiel: BEEP 10
BEEP A

Das Signal ertönt sooft, wie es der ganzzahlige Anteil des Ausdrucks angibt.

4. DIE KOMMENTARVEREINBARUNG REM

Die Kommentarvereinbarung gestattet es, Programme mit Hinweisen zu versehen, die bei einer späteren Überprüfung des Programms dem besseren Verständnis dienen. Vorallem bei Unterprogrammen ist ein derartiger Hinweis angezeigt. Die Kommentarvereinbarung ist keine Anweisung, die dem Fortgang des Programms dient. Sie wird beim Programmlauf übersprungen und erscheint lediglich beim Auflisten des Programms. Die allgemeine Form der Kommentarvereinbarung lautet:

REM Kommentar

Beispiel: 80 : REM *UNTERPROGRAMM 1*

5. KOMMANDOANWEISUNGEN

Zusätzlich zu den programmierbaren Anweisungen kennt der Rechner Kommandoanweisungen, die für den Start eines Programms, für die Programm-Ausführung, Unterbrechung und Auflistung bestimmt sind. Anschließend an die Kommandoanweisungen muß ausnahmslos die Taste ENTER gedrückt werden. Die Anweisung wird daraufhin sofort ausgeführt.

5.1 Programmstart mit RUN

Die Anweisung RUN kann in den Betriebsarten RUN und DEF verwendet werden. Sie veranlaßt den Start des Programmlaufs. Folgende allgemeine Formen sind vorgesehen:

(1) RUN ENTER

Diese Anweisung startet das Programm beginnend mit der Zeile mit der niedrigsten Zeilennummer.

(2) RUN Ausdruck ENTER

Beispiel: RUN 30 ENTER
RUN A(9) ENTER

Diese Anweisung startet das Programm beginnend mit der Zeile, die dem ganzzahligen Anteil des Ausdrucks entspricht. Dieser Wert muß im Bereich 1 bis 999 liegen, sonst erscheint Fehlermeldung 2. Im obigen Beispiel beignnt der Programmlauf mit Zeile 30 bzw. der in Speicher A(9) stehenden Zahl. Steht die gewünschte Zeile nicht im Programm, meldet der Rechner ebenfalls Fehler 2.

(3) RUN { "Text" / Zeichenvariable } ENTER

Beispiele: RUN "PRO-1" ENTER
RUN A$ ENTER

Diese Anweisung startet ein Programm mit Programmnamen. Der Programmname ist entweder durch die ersten 7 Zeichen des eingetasteten Textes oder durch den der Zeichenvariablen zugeordneten Text bestimmt. Steht der gewünschte Programmname nicht im Programmregister, meldet das der Rechner mit Fehlercode 2.

5.2 Programmüberprüfung mit DEBUG

Die in den Betriebsarten RUN und DEF verwendbare Kommandoanweisung DEBUG wird in den gleichen allgemeinen Formen wie RUN eingegeben. Im Gegensatz zu RUN läuft das Programm aber nicht automatisch ab, sondern kann Zeile für Zeile überprüft werden. Diese Anweisung ist also besonders für die Fehlersuche nützlich. Nach Eintasten der DEBUG-Anweisung steht in der Anzeige die Nummer der angewählten Zeile. Andauernder Druck auf ↑ zeigt den Text der Zeile; Druck auf ↑ wählt die nächstfolgende Zeile. Nähere Einzelheiten findet man im Kapitel V7.

5.3 Programmfortführung mit CONT

Die Kommandoanweisung CONT ist in den Betriebsarten RUN und DEF wirksam. Sie startet den automatischen Programmlauf nach einer Programmunterbrechung. Gründe für diese Unterbrechung können sein:

An der betreffenden Stelle steht im Programm STOP.
Der Programmlauf wurde manuell mit der Taste ON/BREAK unterbrochen.
Schrittweises Abarbeiten eines Programms im DEBUG-Betrieb.

Stopt der Programmlauf nach einer mit PRINT programmierten Datenanzeige, kann man ihn anstatt allein mit ENTER auch mit der Kommandoanweisung CONT starten.

Die allgemeine Form des Programmfortführungskommandos lautet:

CONT ENTER

Die Arbeitsweise der Anweisung CONT beim Überprüfen eines Programms zeigt Programm 27.

```
10 : A=0
20 : FOR  B=1 TO  3
30 : A=A+B : PAUSE  B, A
40 : NEXT  B
50 : END
```

PROGRAMM 27

Eingabe	Anzeige			Anmerkung
"RUN"				Betriebsart RUN.
D E B U G	DEBUG_			Befehl zur Überprüfung.
ENTER	10:			
↓	20:			
↓		1.	1.	
	30:			
↓	40:			
C O N T	CONT_			Ende der Überprüfung mit CONT.
ENTER		2.	3.	Automatischer Programmlauf.
		3.	6.	

5.4 Auflisten der Programme mit LIST

Die Kommandoanweisung LIST gestattet es, in der Betriebsart PRO Programme aufzulisten und zu überprüfen. Die Anweisung läßt sich in drei allgemeinen Formen verwenden:

(1) LIST ENTER

Diese Anweisung zeigt die Programmzeile mit der niedrigsten Zeilennummer.

(2) LIST Ausdruck ENTER

Beispiel: LIST 10 ENTER

Diese Anweisung zeigt die dem Wert des Ausdrucks entsprechende Zeilennummer. Der Rechner verwendet allein den ganzzahligen Anteil dieses Wertes. Es sind Zahlen von 1 bis 999 erlaubt. Steht die gewünschte Zeile nicht im Programm, meldet der Rechner Fehler 2.

(3) LIST { **"Text"** / **Zeichenvariable** } ENTER

Beispiel: LIST "PRO-1" ENTER

Diese Anweisung zeigt die Programmzeile, in der der gesuchte Programmname steht. Der Programmname ist entweder durch die ersten 7 Zeichen des eingetasteten Textes oder durch den der Zeichenvariablen zugeordneten Text bestimmt. Steht der gewünschte Programmname nicht im Programmregister, meldet das der Rechner mit Fehlercode 2.

Hat man mit einer der Formen (1) bis (3) die gewünschte Zeile gewählt, kann man die nachfolgende Zeile mit der Taste ↓ anwählen; die vorausgehende bei (2) und (3) mit der Taste ↑ . Haben Programmzeilen mehr als 24 Zeichen, können sie nicht auf einmal angezeigt werden. In diesem Fall werden sie beginnend mit dem ersten Zeichen der Zeile geschrieben. Der nicht sichtbare Teil kann mit Hilfe der Taste ► in die Anzeige gerollt werden.

Das Auflisten soll am Beispiel von Programm 27 gezeigt werden.

Eingabe	Anzeige	Anmerkung
"PRO"	>	Betriebsart PRO.
[L][I][S][T]	LIST_	
[ENTER]	10:A=0	Anzeige der ersten Zeile.
[L][I][S][T]30	LIST30_	
[ENTER]	30:A=A+B:PAUSE B,A	Anzeige von Zeile 30.

5.5 Löschen mit NEW

Das Löschkommando NEW wird in der Form

NEW [ENTER]

verwendet. Im Anschluß an die erfolgte Löschung erscheint das Bereitschaftssymbol. NEW hat in verschiedenen Betriebsarten unterschiedliche Bedeutung.

5.51 NEW in den Betriebsarten DEF, RUN und PRO

In diesen Betriebsarten bewirkt die Kommandoanweisung NEW die Löschung aller Daten- und Programmregister. Allein der Inhalt des RESERVE-Registers bleibt gespeichert.

5.52 NEW in der Betriebsart RESERVE

In Betriebsart RESERVE bewirkt die Anweisung NEW allein die Löschung des RESERVE-Registers. Der Inhalt aller übrigen Register bleibt gespeichert.

5.6 Abfrage des verfügbaren Speicherplatzes mit MEM

Mit der Kommandoanweisung

MEM [ENTER]

erfragt man die noch verfügbare Anzahl von Programmschritten und Speicherplätzen des flexiblen Speichers.

Beispiel: Nach dem Löschen der Programm- und Datenregister sind beim PC-1212 1424 Programmspeicherplätze oder 178 flexible Speicher verfügbar. Die 26 Basisspeicher sind in der angegebenen Anzahl nicht enthalten.

Eingabe	Anzeige
"PRO"	
[N][E][W][ENTER]	>
[M][E][M][ENTER]	1424STEPS 178MEMORIES

Ein Speicherplatz des flexiblen Speichers belegt 8 Programmschritte. Dementsprechend wird durch ein Programm bis zu 8 Schritten die Anzahl der flexiblen Speicher um 1 reduziert; bei 9 bis 16 Schritten stehen 2 Speicher weniger zur Verfügung u.s.w. Je länger also ein Programm, desto geringer die Anzahl der flexiblen Speicher.

6. KOMMANDOTASTEN

6.1 Löschtasten [ON/CA] [CL]

Beide Tasten löschen den gesamten Inhalt des Eingaberegisters in allen 4 Betreibsarten. Fehlermeldungen werden ebenfalls gelöscht.

6.2 Manuelle Programmunterbrechung [BRK]

Möchte man den Programmlauf manuell unterbrechen, verwendet man die ebenfalls der Einschalttaste zugeordnete Funktion [BRK].

Hat man eine Endlosschleife der Form

```
10 : A = 0
20 : A = A + 1 : PAUSE A
30 : GOTO 20
```

PROGRAMM 28

so ist die [BRK]-Taste die einzige Möglichkeit, um den Programmlauf wieder zu stoppen. Der Rechner meldet sich in diesem Fall mit

BREAK AT Zeilennummer

6.3 Eingabetaste [ENTER]

Die Taste [ENTER] gibt den Inhalt des Eingaberegisters an die Rechen- Daten- und Programmregister weiter. Sie ist daher am Ende jeder Eingabe zu bedienen.

IX FEHLERBEHANDLUNG

Der Rechner erkennt Syntaxfehler, also formale Verstöße gegen die BASIC-Regeln und Ablauffehler. Ein Ablauffehler liegt etwa vor, wenn ein Programmsatz zur Suche einer bestimmten Zeile zwar formal richtig geschrieben ist, die gesuchte Zeile aber nicht existiert. Angezeigt wird ein Fehler durch eine Reihe von Punkten und Fehlercode 1 bis 6.

1. FEHLER BEIM MANUELLEN RECHNEN

Beispiel:

Eingabe	Anzeige
"RUN"	
5 [+] [*] 3	5+*3_
[ENTER]	1. . . . .
[◄]	5+*3

Bei diesem Beispiel liegt ein Syntaxfehler vor, da in einem BASIC-Satz [*] nicht auf [+] folgen darf. Drückt man [►] oder [◄] wird die fehlerhafte Eingabe in die Anzeige zurückgerufen und der Cursor steht an der Stelle, an der ein Fehler erkannt wurde, also beim Rechenbefehl *.

Die Fehleranzeige kann man mit einer der Tasten [ON/CA], [CL], [►] oder [◄] zum Verschwinden bringen. Verwendet man [ON/CA] oder [CL], so wird die gesamte Eingabe gelöscht. Die Tasten [►] und [◄] dagegen holen die Eingabe in die Anzeige. Die fehlerhafte Stelle wird durch den Cursor markiert. Korrektur ist durch Überschreiben, Einfügen oder Löschen von Zeichen möglich. Nach der Korrektur veranlaßt man die Ausführung der Rechnung mit [ENTER].

2. PROGRAMMFEHLER

Beispiel: Die Zeile 30 eines Programms lautet 30 : A = 5 + * 3, enthält also einen Syntaxfehler. Der Programmlauf wird in diesem Fall in Zeile 30 unterbrochen. In der Anzeige erscheint die Nummer der Zeile, in der der Rechner den Fehler entdeckt hat. Mit der Taste [↑] kann man den Inhalt der fehlerhaften Zeile abrufen und kontrollieren. Läßt man die Taste wieder los, erscheint das Bereitschaftssymbol.

Eingabe	Anzeige
	>
"RUN"	
[R] [U] [N] [ENTER]	30 : 1. . . . .
[↑]	30 : A=5+*3
	>

Die Fehleranzeige kann man mit den Tasten [0%/d] , [CL] oder [↑] löschen. Lediglich Fehler, die im Zusammenhang mit dem Befehl CHAIN auftreten, lassen sich nicht mit [↑] löschen. Zur Korrektur des Programmfehlers wählt man im Anschluß an die Löschung der Fehlermeldung die Betriebsart PRO und bedient eine der Tasten [↑] oder [↓] . In der Anzeige erscheint die fehlerhafte Programmzeile. Die vom Rechner gefundene Fehlerstelle ist mit dem Cursor markiert. Wie im manuellen Rechenbetrieb wird nun korrigiert und anschließend mit [ENTER] die korrigierte Programmzeile wieder ins Programmregister gelesen.

3. FEHLERCODE UND FEHLERURSACHEN

Fehler code	Ursache	Der Fehler tritt in folgenden Fällen auf
1	Syntaxfehler	Der Betrag eines Ergebnisses übersteigt 10^{100}.
	Zahlenüberlauf	Eine mathematische Funktion ist für das gewählte Argument nicht definiert.
	Fehler beim Speichernamen	Eine Zeichenvariable wird aufgerufen der zugehörige Speicher enthält aber eine Zahl u.u.
2	Zeilenfehler	Zeilennummern oder Marken, die mit den Anweisungen GOTO, GOSUB, RUN, DEBUG oder LIST gesucht werden, existieren nicht.
3	Schleifen- und Unterprogrammfehler	Unterprogramme (GOSUB-RETURN) oder Schleifen (FOR-NEXT) werden in mehr als 4 Ebenen programmiert.
		Der Befehl RETURN wurde ohne vorausgehendes GOSUB geschrieben.
		Der Befehl NEXT wurde ohne vorausgehendes FOR geschrieben.
4	Speicherüberlauf	Die Schrittanzahl eines Programms oder eines RESERVE-Ausdrucks übersteigt die verfügbaren Programmschritte.
		Es wurden indizierte Variablen gewählt, deren Platz bereits von einem Programm besetzt ist.
5	Fehler beim Rekorderbetrieb	Bei der Überprüfung der auf Magnetband geschriebenen oder in den Rechner eingelesenen Programme und Daten werden Fehler entdeckt.
6	Formatfehler	Die errechnete Zahl kann nicht in dem von der USING-Anweisung geforderten Format dargestellt werden.

X DATENSPEICHERUNG AUF MAGNETBAND

Die Computer PC-1212 können gewöhnliche Kassettenrekorder oder Tonbandgeräte als externe Speicher verwenden. Dafür benötigt man als Zusatzgerät das Bandinterface CE-121. Mit ihm kann man den Inhalt aller Programm- und Datenregister des Rechners auf Band speichern und umgekehrt vom Band in den Rechner einlesen. Weiter ist es zur Kontrolle möglich, den Bandinhalt mit den im Rechner gespeicherten Informationen zu vergleichen.
Gibt man den auf das Band gelesenen Informationen, also den Programmen, RESERVE-Ausdrücken und Daten in den Speichern Blocknamen, sucht der Computer die angeforderten Blöcke beim Auslesen aus dem Rekorder automatisch.
Auf diese Weise kann man sich eine leistungsfähige Programmbibliothek auf Band schaffen ohne daß man die Programme jedesmal mühsam mit der Hand eintasten muß.

1. ANFORDERUNGEN AN REKORDER ODER TONBANDMASCHINE

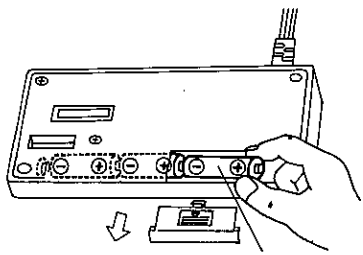
Das magnetische Aufzeichnungsgerät muß folgende Eingenschaftenund Daten besitzen, die jedoch von den meisten handelsüblichen Geräten erbracht werden:

(1) Mikrofoneingang mit einer Eingangsimpedanz kleiner 1 kΩ und einer Empfindlichkeit von mindestens 3 mV.
(2) Lautsprecher, Kopfhörer oder Monitor-Ausgang mit einem Widerstand kleiner 10Ω und einer Ausgangsspannung größer 1 V.
(3) Wünschenswert aber nicht unbedingt erforderlich ist ein Fernbedienungsanschluß.
(4) Der Klirrfaktor bei Aufnahme und Wiedergabe muß kleiner als 15% im Frequenzbereich von 2 bis 4 kHz sein.
(5) Sollten die Eingänge Ihres Bandgerätes nicht mit den Anschlüssen des Interface CE-121 zusammenpassen, müssen im Rundfunkgeschäft erhältliche Adapter verwendet werden.

2 ANSCHLUSS DES RECHNERS PC-1212 AN DAS INTERFACE CE-121

2.1 Einsetzen der Batterien in das Interface

Die Batterien des CE-121 liefern die Energie für die Fernbedienung des Bandgerätes. Sind die Batterien verbraucht, arbeitet die Fernbedienung auch dann nicht mehr, wenn alle Verbindungen ordnungsgemäß hergestellt sind. In diesem Fall ersetzt man *alle drei* leeren Batterien durch neue 1,5V-Batterien der Größe AA. Wie im folgenden Bild gezeigt, muß man auf die richtige Polung der Batterien achten.



3 Stück SUM-3(E) (AA oder R6)

Etwa ein Jahr nach dem Kauf der Batterien sollte man ihren Zustand überprüfen, um Schäden durch Auslaufen der Batterien zu vermeiden. Läßt man eine undichte Batterie im Gerät, kann das nicht mehr reparierbare Schäden verursachen.

2.2 Verbinden des Computers mit dem Interface

Den Rechner schließt man auf folgende Weise an das Interface an:

(1) Der Rechner wird ausgeschaltet.

(2) Man entfernt die Schutzabdeckung seitlich links am Rechner, um die Anschlußleiste freizulegen (Abb. 1).

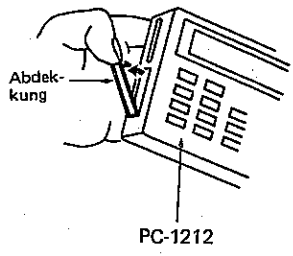


Abb. 1

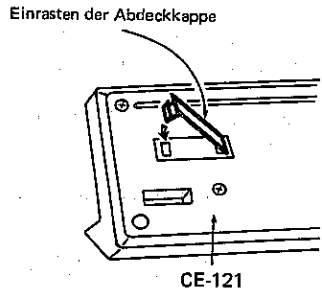


Abb. 2

(3) Um die kleine Abdeckkappe nicht zu verlieren, befestigt man sie in der dafür vorgesehenen Aussparung an der Rückseite des Interface (Abb. 2).

(4) Man legt den Rechner wie in Abb. 3 gezeigt auf das Interface und zwar so, daß sich die linke Kante des Rechners auf der Höhe des roten Pfeils am Innenboden des Interface befindet.

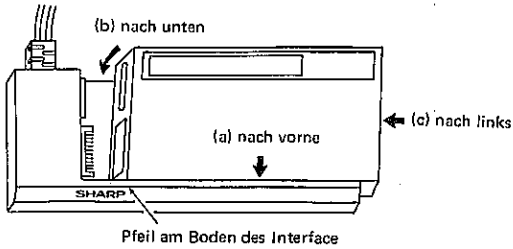


Abb. 3

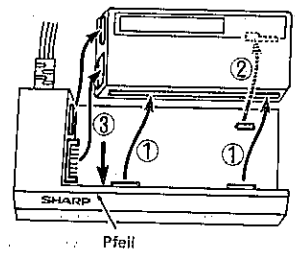


Abb. 4

(5) Nun schiebt man den Rechner in Pfeilrichtung (a) nach vorne, sodaß die Führungsstege ① des Interface (Abb. 4) in die Nuten des Rechners einrasten.

(6) Anschließend drückt man die Oberseite des Rechners *ohne Gewaltanwendung* in Pfeilrichtung (b) nach unten, sodaß die Führung ② des Interface in die entsprechende Nut des Rechners einrastet. Sollte sich ein Widerstand zeigen, verschiebt man den Rechner gegenüber der vom Pfeil markierten Position geringfügig nach rechts oder links, bis er sich leicht nach unten drücken läßt.

(7) Zum Abschluß schiebt man den Rechner, der jetzt satt auf dem Interface aufliegen muß, in Richtung (c) nach links bis zum Anschlag, sodaß die Steckerleiste des Interface mit der Kontaktleiste ③ des Rechners verbunden wird. Die Einheit ist jetzt betriebsbereit.

(8) Die Entfernung des Rechners aus dem Interface geschieht in umgekehrter Reihenfolge.

Man achte unbedingt darauf, daß der Rechner beim Verbinden mit dem Interface und beim Lösen dieser Verbindung mit OFF abgeschaltet ist.

Wenn man entgegen diesem Hinweis den eingeschalteten Rechner mit dem Interface verbindet oder vom Interface löst, ist es möglich, daß anschließend alle Tasten außer Funktion sind. In diesem Fall muß man die Taste ALL RESET am Boden des Computers drücken. Damit werden alle gespeicherten Instruktionen ausnahmsols gelöscht und der Rechner ist wieder betriebsbereit.

3. ANSCHLUSS DES INTERFACE CE-121 AN DAS BANDGERÄT

Bandgerät und Interface werden über 3 Stecker mit folgender farblicher Kennzeichnung und Funktion verbunden:

1) Roter Stecker "MIC"
2) Grauer Stecker "EXT. SP."
3) Schwarzer Stecker "REMOTE"

Unter Beachtung der in Kapitel X-1 beschriebenen Bedingungen werden die Stecker wie folgt angeschlossen:

1) Der rote Stecker des CE-121 wird mit dem Mikrofoneingang des Bandgeräts verbunden. Der Der Mikrofoneingang kann verschieden gekennzeichnet sein, gebräuchlich sind "MIC", "MICRO" oder das DIN-Symbol.
   Ungeeignet sind die mit AUX bezeichneten Eingänge hoher Impedanz. Bei Stereogeräten wird entweder der linke oder rechte Kanal verwendet.
   Bei Mikrofoneingängen mit 6,3 mm-Buchse muß ein handelsüblicher Zwischenstecker verwendet werden.

2) Der graue Stecker des CE-121 wird mit Lautsprecherausgang verbunden. Der Lautsprecherausgang kann verschieden gekennzeichnet sein, gebräuchlich sind "EXT. SP.", "EAR-PHONE" oder das DIN-Symbol. Bei Stereogeräten müssen Eingang (roter Stecker) und Ausgang (schwarzer Stecker) dem selben Kanal zugeordnet sein.
   Bei Lautsprecherausgängen, die nicht zum grauen Stecker passen, muß ein Zwischenstecker verwendet werden.

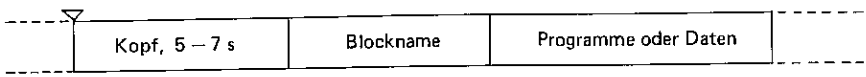
3) Besitzt Ihr Bandgerät eine Fernbedienung, so können Sie das Band vom Rechner gesteuert starten und stoppen. Dazu steckt man den schwarzen Stecker des CE-121 in den Fern bedienungseingang des Bandgerätes. Der Fernbedienungseingang kann verschieden gekennzeichnet sein, gebräuchlich sind "REMOTE", "REM." oder "R/C".
   Bei Fernbedienungseingängen, die nicht zum schwarzen Stecker passen, muß ein Zwischenstecker verwendet werden.
   Sobald man diese Verbindung hergestellt hat, läßt sich das Bandgerät im allgemeinen nicht mehr von Hand, sondern nur mehr computergesteuert bedienen. Möchte man daher etwa für den schnellen Vor- und Rücklauf manuell arbeiten, muß man den schwarzen Stecker aus dem Fernbedienungseingang ziehen.

4) Besitzt das Bandgerät keine Fernsteuerung, ist eine Bandsteuerung vom Computer aus nicht möglich. Der dazugehörige Stecker bleibt unbenutzt und das Bandgerät wird manuell gestartet und gestoppt.

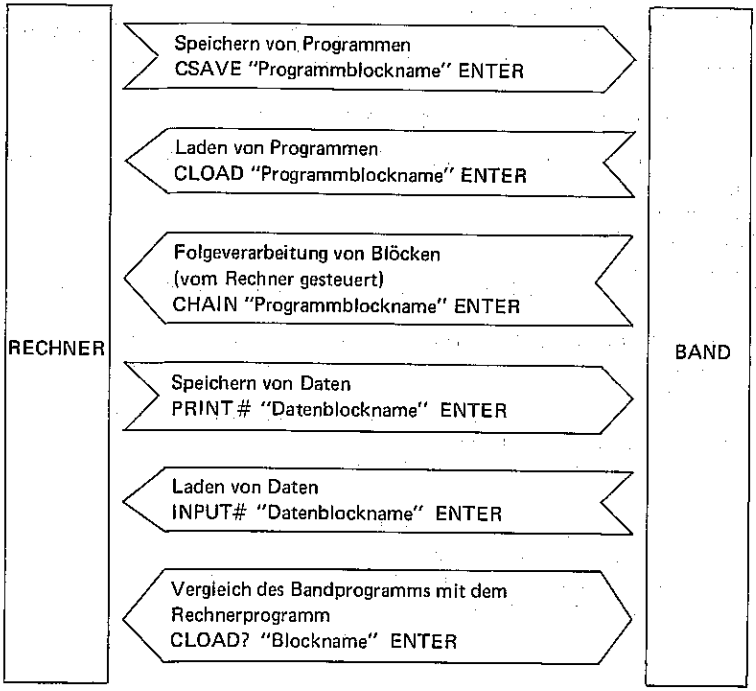
4. ANWEISUNGEN FÜR DEN DIALOG RECHNER – BAND

Programme und Daten werden in Blöcken auf Magnetband geschrieben. Jeder Block beginnt mit einem Blocknamen, der aus maximal 7 Zeichen bestehen darf und in Anführungszeichen stehen muß. Anschließend kommen das Programm, der RESERVE-Ausdruck oder die Daten. Abgeschlossen wird der Block, wenn alle Programm oder Daten auf Band gespeichert sind. Um eine sichere Trennung der Blöcke zu gewährleisten, schreibt der Rechner vor dem Blocknamen automatisch etwa 5 bis 7 Sekunden lang einen konstanten Signalton. Ein kompletter Block hat damit folgende Form:

Start mit ENTER



Folgende Möglichkeiten werden beim Bandbetrieb geboten:



4.1 Speichern von Programmen und RESERVE-Ausdrücken CSAVE

Mit der Anweisung CSAVE werden Programme und RESERVE-Ausdrücke auf Band geschrieben. *Die Speicherung ist nur manuell möglich.* Man verwendet die allgemeine Form

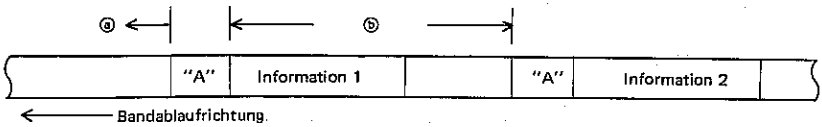
CSAVE "Blockname" ENTER

Das Überspielen dauert abhängig von der Blocklänge einige Minuten. Ist der Block vollständig eingelesen, erscheint das Bereitschaftssymbol in der Anzeige. Stehen keine Instruktionen in den Registern des Rechners, erscheint das Symbol sofort.

Im Anschluß an das Speichern des Blocks sollte man mit der Anweisung CLOAD? überprüfen, ob nicht etwa bedingt durch Bandfehler Information verloren gegangen ist (Abschnitt 4.3).

Wird ein neues Programm so auf Band gespeichert, daß es einen alten Block auch nur etwa überlappt, wird die ursprüngliche Information teilweise gelöscht, was zu einem Fehler beim Laden des alten Blocks führt.

Auf einer Bandseite darf man nie zwei verschiedene Informationen mit gleichem Blocknamen versehen. Haben zwei Blöcke den gleichen Namen, etwa "A", lädt der Rechner das zuerst aufgefundene Programm. Bei dem im Diagramm dargestellten Beispiel wird Block 1 gewählt, wenn der Tonkopf bei Bandstart im Bereich ⓐ steht; befindet er sich jedoch im Bereich ⓑ, wird Block 2 gewählt.



4.11 Programmspeicherung

Das Speichern von Programmen ist in den Betriebsarten RUN, PRO und DEF möglich. Die Speicheranweisung schreibt zuerst den gewählten Programmblocknamen, im Beispiel "PROG-1", und anschließend den *gesamten* Inhalt des Programmregisters auf Band.

Beispiel: Betriebsart PRO; CSAVE "PROG-1" ENTER

4.12 Speichern von RESERVE-Ausdrücken

RESERVE-Ausdrüke werden allein in der Betriebsart RESERVE gespeichert. Die Speicheranweisung schreibt den gewählten RESERVE-Blocknamen und anschließend *alle* RESERVE-Ausdrücke auf Band.

Beispiel: Betriebsart RESERVE; CSAVE "RESRV-1" ENTER

Zweckmäßigerweise legt man für jedes Band einer Programmbibliothek ein Karteiblatt mit den Programmnamen und den dazugehörigen Programmen an. Hat man die Programmnamen vergessen, ist es nicht mehr möglich, die Informationen in den Rechner zu laden. Um das Laden der Programme zu beschleunigen sollte man auch die Lage des gesuchten Blocks auf dem Band, eventuell die Nummer des Zählwerks, notieren.

4.2 Laden von Programmen und RESERVE-Ausdrücken CLOAD

Mit der Ladeanweisung CLOAD werden Programme und RESERVE-Ausdrücke vom Rechner gesucht und vom Band in den Rechner geladen. *Das Laden ist nur manuell möglich*; der Blockname wird aber automatisch gesucht. Vor dem Laden sollte man das Band soweit umspulen, daß der Tonkopf *vor* dem gesuchten Block steht. Die allgemeine Form der Ladeanweisung lautet:

CLOAD "Blockname" ENTER

Der Rechner kann nicht unterscheiden, ob ein Blockname zu einem Programm oder zu einem RESERVE-Ausdruck gehört. Bei falscher Wahl der Betriebsart besteht daher die Gefahr, daß RESERVE-Ausdrücke in das Programmregister und Programm in das RESERVE-Register geladen werden.

Befindet sich der gesuchte Blockname *nicht* auf dem eingelegten Band, sucht der Computer auch dann nach dem fehlenden Namen weiter, wenn das Band abgelaufen ist. In diesem Fall muß man den Ladebefehl mit ON/CA löschen. Das gilt auch für die in den folgenden Abschnitten beschriebenen Anweisungen CLOAD?, CHAIN und INPUT#.

Tritt beim Laden eines Programms ein Fehler auf, wird allein das Programmregister gelöscht. Das gilt auch für die später beschriebene Anweisung CHAIN.

4.21 Laden von Programmen

Programme lassen sich in den Betriebsarten RUN, PRO und DEF vom Band in den Rechner laden. Die Ladeanweisung sucht zuerst den gewählten Blocknamen, im Beispiel "PROG-1", und lädt anschließend den gesamten Block.

Beispiel: Betriebsart PRO; CLOAD "PROG-1" ENTER

4.22 Laden von RESERVE-Ausdrücken

RESERVE-Ausdrücke werden allein in der Betriebart RESERVE geladen. Die Ladeanweisung sucht zuerst den gewählten Blocknamen des RESERVE-Ausdrucks, im Beispiel "RESRV-1", und lädt anschließend den gesamten Block.

Beispiel: Betriebsart RESERVE; CLOAD "RESRV-1" ENTER

4.3 Vergleich der Bandinformation mit der Rechnerinformation CLOAD?

Mit der Testanweisung CLOAD? vergleicht man die auf Band gespeicherte Information, also Programme und RESERVE-Ausdrücke, mit der Information gleichen Blocknamens im Computer. *Der Vergleich kann nur manuell abgerufen werden.* Werden Abweichungen festgestellt, meldet das der Rechner mit dem Fehlercode 5. In diesem Fall liest man das überprüfte Programm nochmals auf Band und vergleicht im Anschluß wieder. Sind in den Registern des Rechners keine Informationen gespeichert, meldet sich der Rechner sofort mit dem Bereitschaftssymbol. Auch beim Informationsvergleich sollte man den Tonkopf des Bandgerätes in der Nähe des Blockanfangs positionieren.
Die allgemeine Form des Informationsvergleichs lautet:

CLOAD? "Blockname" ENTER

4.31 Überprüfen von Programmen

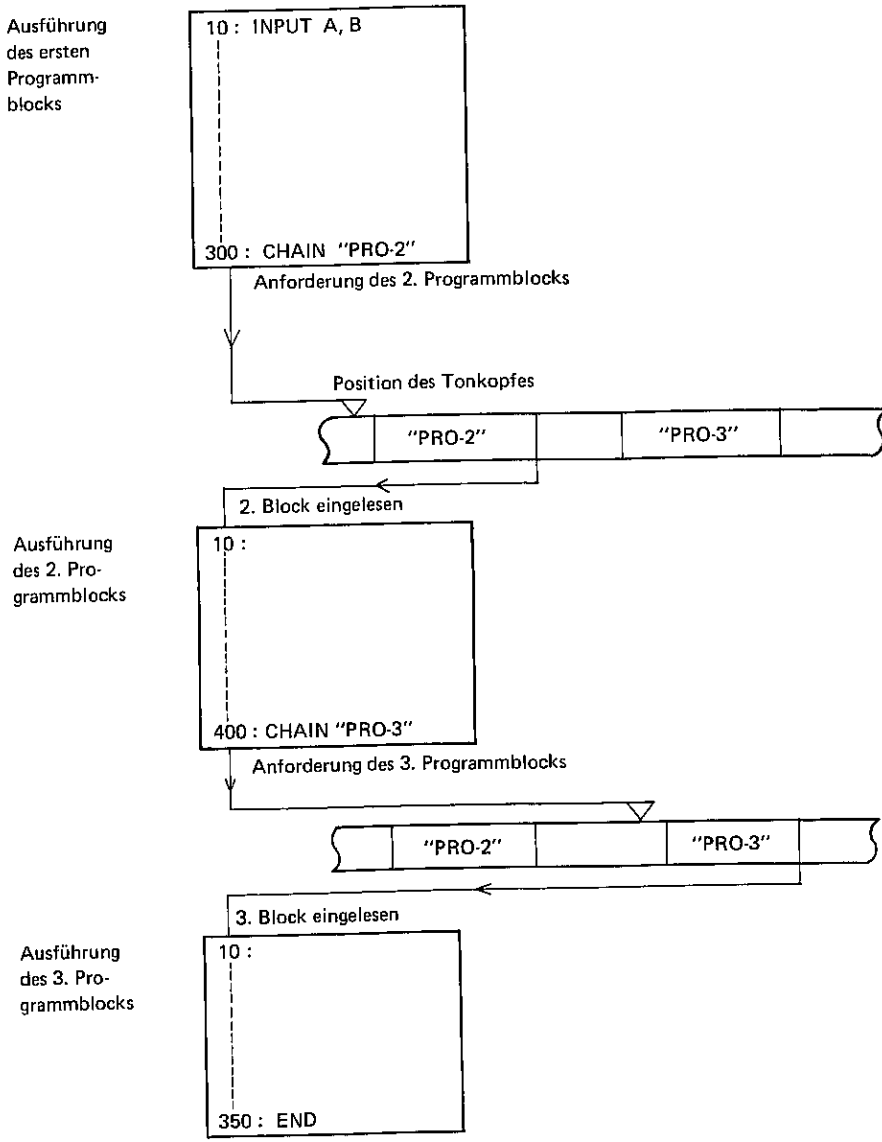
Programme lassen sich in den Betriebsarten RUN, PRO und DEF überprüfen. Verglichen wird das Programm auf dem Band mit dem im Programmregister gespeicherten Programm gleichen Namens.

4.32 Überprüfen von RESERVE-Ausdrücken

RESERVE-Ausdrücke können allein in der Betriebsart RESERVE überprüft werden. Verglichen werden die RESREVE-Ausdrücke auf Band mit den im RESERVE-Register gespeicherten Ausdrücken gleichen Namens.

4.4 Rechnergesteuerte Anforderung von Bandprogrammen CHAIN

Steht in einem Programm die Anweisung CHAIN mit Blocknamen, such der Rechner automatisch das angeforderte Programm am Band, lädt es ins Programmregister und führt es anschließend beginnend bei einer spezifizierten Zeilennummer aus. Auf diese Weise kann man lange lineare Programme welche die Kapazität des Programmregisters überschreiten, in Blöcke aufteilen und Block für Block automatisch ausführen. Man erreicht so eine Vervielfachung der Kapazität des Programmregisters. Die Zusammenfügung der Programmblöcke veranschaulicht das folgende Diagramm.


Ausführung des ersten Programm-blocks
10 : INPUT A, B
300 : CHAIN "PRO-2"
Anforderung des 2. Programmblocks
Position des Tonkopfes
"PRO-2"
"PRO-3"
2. Block eingelesen
Ausführung des 2. Pro-grammblocks
10 :
400 : CHAIN "PRO-3"
Anforderung des 3. Programmblocks
"PRO-2"
"PRO-3"
3. Block eingelesen
Ausführung des 3. Pro-grammblocks
10 :
350 : END

Man sieht, daß es zunächst notwendig ist, den Tonkopf *vor* das erste angeforderte Programm zu bringen. Alles weitere steuert der Rechner selbsttätig. Der CHAIN-Befehl in Zeile 300 des ersten Blocks ruft den nächsten Block ab, lädt ihn in den Rechner, arbeitet ihn ab und fordert in Zeile 400 Block 3 an. Diese Verkettung von Blöcken kann man im Rahmen der Bandkapazität beliebig oft wiederholen.

Der Rechner kennt drei Möglichkeiten, Programm zu verketten:

(1) CHAIN "Blockname"

Mit diesem Befehl wird das gewählte Programm geladen und beginnend bei der ersten Zeile ausgeführt.

Beispiel: 100 : CHAIN "ABC"

In Zeile 100 wird das Programm "ABC" gesucht, vom Band in den Rechner geladen und von Anfang an ausgeführt.

(2) CHAIN "Blockname", Ausdruck

Auch dieser Befehl lädt das gewählte Programm. Die Ausführung beginnt aber bei der durch den Ausdruck bestimmten Zeilennummer. Als Zeilennummer verwendet der Rechner den ganzzahligen Anteil des Ausdrucks; Zeilennummern von 1 bis 999 sind zulässig.

(3) CHAIN "Blockname", { **"Text"** / **Zeichenvariable** }

Bei dieser Anweisung beginnt der Programmlauf nach dem Laden in der der durch eine Marke gekennzeichneten Zeile. Als Marke dient der "Text", der entweder in Klarschrift im Programm steht oder in dem der Zeichenvariablen zugeordneten Speicher zu finden ist.

Beispiel: 300 : CHAIN "XYZ", "MATH-1"

In Zeile 300 wird das Programm "XYZ" gesucht, vom Band in das Programmregister geladen und beginnend mit der durch die Marke "MATH-1" markierten Zeile ausgeführt.

4.5 Speichern von Daten PRINT

Mit der Anweisung PRINT# werden die Inhalte der Datenspeicher auf Band geschrieben. Diese Speicherung kann in den Betriebsarten RUN und DEF *sowohl manuell als auch rechnergesteuert* erfolgen. Bei manueller Speicherung muß direkt auf die Speicheranweisung folgend die Taste [ENTER] gedrückt werden. Der Rechner kann Datenblöcke von Programmblöcken unterscheiden. Es ist daher möglich, die zu einem Programm gehörigen Daten mit dem gleichen Namen zu versehen. Zwei allgemeine Formen sind vorgesehen:

(1) PRINT # "Blockname"

Diese Anweisung versieht zuerst das Band mit dem Blocknamen. Anschließend werden die Inhalte aller Datenspeicher, also der Basisspeicher und der flexible Speicher, beginnend bei A bzw A$ auf Band übertragen.
Beispiel für das manuelle Übertragen von Daten auf Band:

PRINT # "DATEN-1" [ENTER]

Die Anweisung gibt dem Datenblock den Namen "DATEN-1" und speichert im Anschluß alle im Rechner verfügbaren Daten auf Band.

(2) PRINT "Blockname"; Variable

Auch diese Anweisung gibt dem Datenblock den gewünschten Namen. Gespeichert werden aber nur die Daten, die im Speicher mit dem gewählten Variablennamen stehen und alle nachfolgenden. Der Variablenname darf in der Form A bis Z oder A ( ) geschrieben werden. Der Index muß bei der Wahl eines flexiblen Speichers so niedrig sein, daß der Datenspeicherbereich nicht das Programm erreicht, da sonst wie in diesem Fall üblich Fehler 4 gemeldet wird.

Beispiel für rechnergesteuerte Datenspeicherung:

150 : PRINT # "DATEN-1"; A(26)

In Zeile 150 wird die Programmausführung gestoppt. Der Rechner schreibt automatisch den Blocknamen "DATEN-1" auf das Band und speichert alle Daten ab Speicher A(26) = Z.

4.6 Laden von Daten INPUT#

Diese Anweisung gibt den Befehl, einen auf Band gespeicherten Datenblock zu suchen und in die Speicherregister des Rechners zu laden. Dieser Ladevorgang kann in den Betriebsarten RUN und DEF *sowohl manuell als auch rechnergesteuert* erfolgen. Vor dem Laden empfiehlt es sich, das Band kurz vor den Anfang des Datenblocks zu spulen. Entsprechend der PRINT#-Anweisung gibt es zwei allgemeine Formen:

(1) PRINT # "Blockname"

Mit dieser Form wird der Computer angewiesen, die dem Blocknamen zugeordneten Daten zu suchen und in den Datenspeicher des Rechners beginnend mit A oder A$ zu laden.
Beispiel für das manuelle Laden von Daten:

INPUT # "DATEN-1" ENTER

Der Rechner sucht den Block mit dem Namen "DATEN-1" und lädt alle Daten dieses Blocks in den Rechner.

(2) INPUT # "Blockname"; Variable

Diese Anweisung entspricht Form (1), jedoch werden nur die Daten des mit dem Variablennamen gekennzeichneten Speichers und die folgenden geladen. Die Variablennamen sind die gleichen wie bei PRINT# in Kapitel 4.5.

Beispiel für rechnergesteuertes Laden von Daten:

50 : INPUT # "DATEN-1"; A(26)

Der Rechner sucht den Block mit dem Namen "DATEN-1" und lädt die Daten in Speicher A(26) = Z und die folgenden.

5. VORBEREITUNG DER BANDARBEIT

Vor der Verwendung von Band und Interface sind einige Vorbereitungen und von dem Bandgerät abhängige Einstellungen notwendig.

(1) Man stelle vorschriftsmäßig die Verbindung Rechner-Interface-Bandgerät entsprechend den Kapiteln 2 und 3 her.
(2) Rostige oder verbogene Eingangsstecker am Bandgerät sind nicht geeignet.
(3) Man reinige den Tonkopf nach Herstellerangaben.
(4) Man verwende nur Markenbänder mit ausreichendem Frequenzgang.
Die Bänder müssen sauber und frei von Kratzern sein. Bei fehlerhaftem Bandbetrieb eine andere Kassette verwenden.
(5) Für Aufnahme und Wiedergabe sollte man das selbe Gerät verwenden. Ist nämlich der Spalt des Tonkopfes nicht richtig justiert, kann die Verwendung verschiedener Geräte Übertragungsfehler mit sich bringen.
(6) Besitzt Ihr Bandgerät die Bedienungselemente "Höhen" oder "Tiefen", sollten sie zunächst in Mittelstellung gebracht werden. Bei manchen Gerätetypen kann die richtige Justierung dieser Knöpfe Übertragungsfehler vermeiden.
(7) Falls vorhanden, muß die Bandsorteneinstellung entsprechend dem gewählten Band erfolgen.
(8) Bei Bandgeräten mit wählbarer Stromversorgung ist der Batteriebetrieb vorzuziehen. Manche Netzgeräte erzeugen auf dem Band einen so starken Netzbrumm, daß Störungen auftreten können.

(9) Ist die Bandgeschwindigkeit wählbar, sollte man den höheren Wert einstellen.
(10) Gerät mit wählbarer Aussteuerung stellt man auf "AUTO". Ist der Automaticbetrieb nicht vorgesehen, stellt man bei einer Probeaufnahme die optimale Aussteuerung ein.
(11) Bei Stereogeräten verwende man bei Aufnahme und Wiedergabe den selben Kanal. Der Balanceregler wird voll auf den verwendeten Kanal gestellt.
(12) Geräte mit Mischpult sind so einzupegeln, daß der Regler voll auf dem gewählten Mikrofoneingang steht.

6. SPEICHERN VON INFORMATIONEN AUF BAND

Um Bandaufnahmen zu machen, geht man nach folgendem Schema vor:

6.1 Manuelles Überspielen

(1) Die auf Band zu übertragende Information muß im Programmspeicher, im RESERVE-Speicher oder in den Datenspeichern stehen.
(2) Aufsuchen einer für die Aufnahme geeigneten Bandstelle. Ist das Band noch leer, schreibt man die Information in die Nähe des Bandanfangs. Dabei beachte man, daß Bandanfang und Bandende aus unmagnetischem, für die Aufnahme nicht geeignetem Material bestehen. Ein ausreichender Sicherheitsabstand von Bandanfang und Bandende ist daher notwendig. Ist das Band teilweise beschrieben, sucht man den freien Platz im Anschluß an das letzte Programm. Dazu stellt man das Bandgerät auf Wiedergabe und läßt es solange laufen, bis die für die Rechnerinformation typischen Knackgeräusche verstummen. Bei der Wahl der Bandstelle muß man das Fernbedienungskabel entfernen, da der Bandlauf sonst nicht per Hand zu steuern ist.
(3) Dokumentation von Blocknamen und Stellung des Bandes oder Anzeige des Bandzählwerks sowie des Inhaltes der gespeicherten Information.
(4) Der rote Aufnahmestecker und der Fernbedienungsstecker des Interface müssen mit dem Bandgerät verbunden sein.
(5) Eingabe der Speicheranweisung entsprechend den Kapiteln 4.1 und 4.5.

Beispiel: Um ein Programm zu speichern, wählt man die Betriebsart RUN und tastet CSAVE "AA" ein. Damit hat man die Anweisung gegeben, das im Programmregister stehende Programm mit dem Namen "AA" zu versehen und auf Band zu speichern. Handelt es sich um Daten, wählt man PRINT# "AA".

(6) Einstellen des Bandgerätes auf "AUFNAHME". Das geschieht bei den meisten Geräten durch gleichzeitiges Drücken der Tasten "WIEDERGABE" und "AUFNAHME". Geräte ohne Fernbedienung laufen bereits jetzt an.
(7) Start der Eingabe mit ENTER . Bei Geräten mit Fernbedienung wird automatisch der Bandlauf gestartet.
(8) Im Anschluß an ENTER ist zunächst etwa 5 bis 7 Sekunden lang ein Ton konstanter Höhe zu vernehmen. Der Kopf des Übertragungsblocks wird geschrieben. Dann ist ein unterbrochener Ton zu hören: Blockname und Informationen werden geschrieben. Die Zeitdauer dieses Abschnitts hängt von der Länge der Information ab und kann mehrere Minuten betragen.
(9) Ist die gewünschte Information auf Band übertragen, erscheint in der Anzeige das Bereitschaftssymbol. Die Information steht nun sowohl am Band als auch in den Registern des Rechners; die fehlerfreie Speicherung kann mit CLOAD? überprüft werden. Geräte mit Fernbedienung schalten mit dem Erscheinen des Symbols > automatisch ab; handbediente Geräte müssen mit der Taste STOP angehalten werden.

6.2 Programmgesteuertes Überspielen.

Programmgesteuertes Überspielen ist nur bei der Speicherung von Daten möglich. Bei Geräten mit Fenbedienung berücksichtigt man die Punkte (1) bis (4) und schaltet das Bandgerät auf Aufnahme. Alles weitere steuert der Rechner selbst.

Besitzt das Bendgerät keine Fernbedienung, muß man zusätzlich zu (1) bis (4) noch den Programmlauf unterbrechen, um das Band zum richtigen Zeitpunkt für die Aufnahme starten zu können. Das erreicht man durch akustisches Signal und Anweisung STOP vor der Anweisung PRINT #.

Beispiel:
```
140 : BEEP 3 ENTER
150 : STOP : PRINT # "ABC"
```

In Zeile 140 erinnert der Rechner durch 3 Signaltöne an die bevorstehende Datenspeicherung und hält in Zeile 150 bei STOP an. Nun schaltet man das Bandgerät auf Aufnahme. Sobald das Band läuft, setzt man das Programm mit CONT ENTER fort. Die Daten werden gespeichert. Ist die Speicherung zuende, muß man das Band von Hand wieder anhalten.

7. LADEN DES RECHNERS VOM BAND

Um Informationen aus dem Band in den Rechner zu laden, geht man nach folgendem Schema vor:

7.1 Manuelles Laden

(1) Aufsuchen des gewünschten Blocks am Band mit Hilfe der Dokumentation (Kapitel 6.1(3)). Der Tonkopf muß genügend weit vor dem Block stehen. Kennt man die Position des Blocks am Band nicht, läßt man den Rechner vom Bandanfang suchen, bis er selbst den gewünschten Block gefunden hat.

(2) Der schwarze Wiedergabestecker und der Fernbedienungsstecker des Interface müssen mit dem Bandgerät verbunden sein.

(3) *Der Lautstärkeregler muß auf Mitte bis Maximum gestellt werden.*

(4) Eingabe der Ladeanweisung entsprechend Kapitel 4.2 und 4.6.

Beispiel: Um ein Programm zu laden, wählt man die Betriebsart RUN und tastet CLOAD "AA" ein. Damit hat man die Anweisung gegeben, den Programmblock "AA" zu suchen und das Programm in den Rechner zu laden. Handelt es sich um Daten, wählt man INPUT # "AA".

(5) Bei Geräten mit Fernbedienung drückt man die Wiedergabetaste des Bandgerätes und anschließend ENTER . Das Band setzt sich automatisch in Bewegung und der Suchlauf beginnt.

(6) Bei Geräten ohne Fernbedienung drückt man zuerst die ENTER -Taste und setzt das Band in Betriebsart Wiedergabe in Bewegung.

(7) Ist der gesuchte Block in den Rechner geladen, erscheint in der Anzeige das Bereitschaftssymbol. Geräte mit Fernbedienung schalten mit dem Erscheinen des Symbols > automatisch ab; handbediente Geräte müssen mit der Taste STOP angehalten werden.

(8) Ist auch bei Bandende das Bereitschaftssymbol nicht in der Anzeige, steht wahrscheinlich der gesuchte Blockname nicht auf dem Band. In diesem Fall bringt man den Rechner mit Hilfe der Taste ON/CA wieder in Betriebsbereitschaft.

7.2 Programmgesteuertes Laden

INPUT# und CHAIN-Anweisungen werden programmgesteuert ausgeführt. Bei fernbedienten Bandgeräten berücksichtigt man die Punkte (1) bis (3) und schaltet das Bandgerät auf Wiedergabe. Alles weitere steuert der Rechner selbst.
Bei Bandgeräten ohne Fernbedienung wird genau wie im Kapitel 6.2 eine BEEP und eine STOP-Anweisung vor den Ladebefehl ins Programm geschrieben und man startet das Band manuell.

7.3 Fehlermeldung beim Laden

Wird während des Ladens des Programms Fehlercode 5 angezeigt, beginnt man den Ladevorgang nochmals von Anfang. Tritt der Fehler bei mehreren Ladeversuchen auf, muß man den Block neu schreiben und neu auf Band, möglichst an einer anderen Stelle, speichern.

Sollte das Speichern und Laden trotz mehrerer Versuche nicht möglich sein, entspricht das Bandgerät wahrscheinlich nicht den Anforderungen des Kapitels 1 und man sollte es mit einem anderen Gerät versuchen.

8. VERGLEICH VON BANDBLOCK MIT RECHNERBLOCK

Hat man ein Programm auf Band überspielt, kann man mit der Anweisung CLOAD? untersuchen, ob es fehlerfrei gespeichert wurde. Diese Überprüfung läßt sich nur manuell ausführen. Der Prüfungsablauf entspricht weitgehend dem beim Laden, also Kapitel 7. Die Informationen werden aus dem Band gelesen, Zeichen für Zeichen mit der Information in den Registern des Rechners verglichen und sobald eine Abweichung festgestellt wird, meldet das der Rechner mit Fehlercode 5.

Beim Überprüfen geht man wie folgt vor:

(1) Wie in Kapitel 7.1 beschrieben verbindet man Rechner mit Bandgerät, stellt den Lautstärkeregler richtig ein und sucht den zu überprüfenden Programmblock. Dieser Block muß vor dem Tonkopf stehen.

(2) Eingabe der Testanweisung entsprechend Kapitel 4.3.

Beispiel: Man wählt Betriebsart RUN und tastet CLOAD? "AA" ein. Damit gibt man die Anweisung zum Aufsuchen des Programmblocks "AA" und dem Vergleich dieses Blocks mit dem im Rechner gespeicherten Programm.

(3) Bei Geräten mit Fernbedienung bedient man die Wiedergabetaste und anschließend ENTER . Bei Bandgeräten ohne Fernbedienung drückt man zuerst die ENTER -Taste und setzt dann das Band in Betriebsart Wiedergabe in Bewegung.

(4) Stimmen Band- und Rechnerinformation überein, erscheint nach vollständigem Überprüfen des Blocks das Bereitschaftssymbol. Unterscheiden sich die Blöcke auch nur in einem Zeichen, wird die Überprüfung abgebrochen und Fehler 5 angezeigt. In diesem Fall sollte man den Testlauf nochmals starten. Erscheint auch dann die Fehlermeldung, ist der Block nicht korrekt auf Band gespeichert und man muß ihn nochmals entsprechend Kapitel 6 speichern.

X. DIE TASTENFUNKTIONEN

Taste	Funktion
CA/BREAK ON	• Einschalten des Rechners. • Unterbrechung des Programmlaufs. • Löschung des Eingaberegisters und der Fehleranzeige.
OFF	• Ausschalten des Rechners.
SHFT	• Abruf der über den Tasten stehenden Doppelfunktionen wie etwa π und $\wedge$. Beispiel: SHFT $\overset{\pi}{\boxed{\uparrow}}$ $\rightarrow \pi$. • In der Betriebsart DEF ruft die Taste SHFT gefolgt vom Programmnamen, also einer der RESERVABLE KEYS A bis SPC , die mit Namen versehenen Programme ab. Beispiel: SHFT A . • In der Betriebsart RESERVE ermöglicht SHFT gefolgt von einer der RESERVABLE KEYS Definition und Abruf von Benutzerfunktionen. Beispiel: SHFT B . • In den Betriebsarten PRO und RUN ruft SHFT gefolgt von einer der RESERVABLE KEYS die zugeordnete Benutzerfunktion ab. Beispiel: SHFT C .
0 bis 9	• Zifferneingabe.
.	• Festlegung der Position des Dezimalpunktes. • Abschluß der Abkürzung von Anweisungen. Beispiel: D. ≙ DEBUG. • Festlegen der Anzahl der Nachkommastellen bei Formatwahl mit USING.
EXP	• Exponenteneingabe beim wissenschaftlichen Format. In der Anzeige wird dieser Befehl E geschrieben.
A bis Z	• Alphatasten zum Schreiben der BASIC-Anweisungen. • Variablennamen, denen die Speicher A bis Z zugeordnet sind.
/	• Divisionsbefehl.
*	• Multiplikationsbefehl.
+	• Festlegen des positiven Vorzeichens einer Zahl; wird meist weggelassen. • Additionsbefehl.
−	• Festlegen des negativen Vorzeichens einer Zahl. • Subtraktionsbefehl.
SHFT ∧	• Befehl zum Schreiben der allgemeinen Exponentialfunktion. • Festlegen des wissenschaftlichen Formats bei der Formatwahl mit USING.
SHFT <	• Zeichen für "kleiner" zur Festlegung der Vergleichsoperatoren <, <= und <>.
SHFT >	• Zeichen für "größer" zur Festlegung der Vergleichsoperatoren >, >= und <>.

Taste	Funktion
[=]	• Zuordnungsbefehl. Die rechts vom Zuordnungszeichen stehenden Daten werden den links davon stehenden Variablen zugeordnet. • Zeichen für "gleich" zur Festlegung der Vergleichsoperatoren =, <= und >=.
[(], [)]	• Klammerbefehl.
[√]	• Quadratwurzel.
[SPC]	• Die Taste [SPC] (space) schreibt Leerzeichen in Programme und Texte. Damit erreicht man eine wesentlich bessere Übersicht. Der Rechenlauf wird durch Leerzeichen im Allgemeinen nicht beeinflußt. Eine Ausnahme machen hier Programmarken und Zeichenvariablen. "PRO-1" wird als eine andere Marke erkannt als "PRO-1".
[SHFT] [:]	• Der Doppelpunkt trennt Programmsätze, die in eine Programmzeile geschrieben werden.
[SHFT] [;]	• Soll eine PRINT-Anweisung zwei oder mehrere Daten direkt nebeneinanderstehend anzeigen, trennt man die zugehörigen Variablen mit einem Strichpunkt. • Bei der INPUT-Anweisung trennt der Strichpunkt den Kommentar von der Variablen. • Bei der PRINT#- und INPUT#Anweisung trennt der Strichpunkt Blocknamen und Variablennamen.
[SHFT] [,]	• Trennung der Ausdrücke bei gleichzeitiger Eingabe mehrerer Ausdrücke. • Sollen bei der PRINT-Anweisung zwei Daten in den zwei Abschnitten der Anzeige stehen, trennt man die zugehörigen Variablen mit dem Komma. • Bei der INPUT-Anweisung trennt das Komma den Kommentar von der Variablen. • Bei der CHAIN-Anweisung trennt das Komma Blocknamen und Ausdruck oder Blocknamen und Programmarke, falls nicht beim Programmanfang sondern bei der durch Ausdruck oder Marke festgelegten Zeile begonnen werden soll.
[SHFT] [#]	• Bei der Formatanweisung USING legt # die Stellenanzahl fest. • Festlegung der Anweisungen PRINT# und INPUT#.
[SHFT] [?]	• Festlegung der Anweisung CLOAD?.
[SHFT] [$]	• Festlegung der Zeichenvariablen.
[SHFT] ["]	• Kennzeichnung einer Zeichenfolge als "Text". • Festlegung von Programmnamen und Marken.
[SHFT] [!] • [SHFT] [%] • [SHFT] [≠] •	Haben keine Funktion, können aber in Kommentaren als Textzeichen verwendet werden.
[MODE]	• Wahl der Betriebsarten DEF, RUN, PRO und RESERVE.
[CL]	• Löscht Eingaberegister, Anzeige und Fehlermeldung.

Taste	Funktion
[►]	• Schiebt den Cursor nach rechts. • Ruft den Inhalt des Eingaberegisters in die Anzeige (Playback). • Ruft den Cursor in die Anzeige, falls er nicht bereits dort steht.
[◄]	• Schiebt den Cursor nach links. • Die anderen Funktionen sind denen der Taste [►] gleich.
[SHFT] [INS]	• Der Befehl [INS] schafft links des Cursors einen freien Platz. Er ist mit dem Zeichen □ markiert, das verschwindet, sobald man ein anderes Zeichen darüber schreibt.
[SHFT] [DEL]	• Der Befehl [DEL] löscht das mit dem Cursor markierte Zeichen oder eine ganze Anweisung, falls der Cursor auf dem ersten Zeichen einer Anweisung steht.
[ENTER]	• Abschluß einer Programmzeile. • Abschluß eines RESERVE-Ausdrucks. • Ausführungsbefehl beim manuellen Rechnen und bei den Kommandoanweisungen. • Programmstart nach der Dateneingabe infolge einer INPUT-Anweisung. • Programmstart nach der Datenanzeige infolge einer PRINT-Anweisung.

Die Tasten [↑] , [↓] und [ON] haben abhängig von der gewählten Betriebsart und dem Betriebszustand unterschiedliche Funktionen.

<table>
<tr><th>Betriebsart</th><th>Betriebszustand</th><th>[↑]</th><th>[↓]</th><th>[ON]</th></tr>
<tr><td></td><td>Rechner ausgeschaltet.</td><td rowspan="2"></td><td rowspan="2"></td><td>Einschalten.</td></tr>
<tr><td rowspan="5">RUN ,

DEF</td><td>Programmlauf.</td><td rowspan="3">Zeitweilige Programmunterbrechung mit BREAK</td></tr>
<tr><td>Anforderung einer Eingabe mit INPUT</td><td rowspan="3">Anzeige der in Bearbeitung befindlichen oder schon abgearbeiteten Zeile durch Festhalten der Taste.</td><td rowspan="2">Ausführung der Programmüberprüfung (DEBUG).</td></tr>
<tr><td>Im Anschluß an eine Datenanzeige mit PRINT.</td></tr>
<tr><td>Bei Programmunterbrechung mit BREAK.</td><td>Abarbeiten der nächsten Zeile.</td><td rowspan="2">Vollständiges Löschen des Eingaberegisters.</td></tr>
<tr><td>Fehlermeldung beim Programmlauf.</td><td>Anzeige der für den Fehler verantwortlichen Zeile durch Festhalten der Taste.</td><td></td></tr>
<tr><td>PRO</td><td colspan="4">Nur für den Fall, daß keine Zeilennummer angezeigt wird, etwa beim Umschalten von einer beliebigen anderen Betriebsart auf PRO.</td></tr>
</table>

Betriebsart	Betriebszustand	↑	↓	ON
PRO	Im Anschluß an eine Datenanzeige mit PRINT.	Anzeige der Zeile, bei welcher der Programmlauf unterbrochen wurde.		Vollständige Löschung des Eingaberegisters.
	Bei Programmunterbrechung mit BREAK.			
	Löschung der Fehleranzeige mit irgendeiner Taste ausgenommen ON.	Anzeige der fehlerhaften Zeile		
	Nur für den Fall, daß eine Programmzeilennummer angezeigt wird.			
	Überprüfen, Schreiben und Ändern des Programms.	Anzeige der vorangehenden Programmzeile.	Anzeige der nächstfolgenden Programmzeile.	Vollständige Löschung des Eingaberegisters.
RESERVE				

Die mit der Taste MODE wählbaren Betriebsarten bieten folgende Möglichkeiten:

	RUN	PRO	RESERVE	DEF
Manuelles Rechnen	X	–	X	X
Programme schreiben	–	X	–	–
Programme anwenden	X	–	–	X
RESERVE-Ausdrücke schreiben	–	–	X	–
RESERVE-Ausdrücke anwenden	X	X	–	–
Namensprogramme schreiben (DEF)	–	X	–	–
Namensprogramme anwenden (DEF)	–	–	–	X

Als Textzeichen, die in Anführungsstrichen stehen, dürfen alle Tasten bis auf ENTER, ON, OFF, ↑, ↓, ►, ◄, CL, INS, DEL, MODE, SHFT und " verwendet werden.

Die RESERVABLE KEYS A, S, D, F, G, H, J, K, L, =, Z, X, C, V, B, N, M und SPC haben im Anschluß an SHFT folgende Bedeutung:

(1) In der Betriebsart DEF werden die Programme gleichen Namens aufgerufen.
(2) In Betriebsart RESERVE werden RESERVE-Ausdrücke geschrieben und überprüft.
(3) In den Betriebsarten PRO und RUN werden die RESERVE-Ausdrücke abgerufen. Ist kein Ausdruck gespeichert, wird der Speichername geschrieben.

Die Austaste OFF hat während der Ausdführung einer Rechnung oder während des Programmlaufs keine Wirkung.

XII LISTE DER FUNKTIONEN UND ANWEISUNGEN

Bei Abkürzungen muß unbedingt der Punkt gesetzt werden. Nach Eingabe der Abkürzungen in die Register, steht dort der vollständige BASIC-Befehl. Man sieht das beim Rückruf der Information.

1. FUNKTIONEN

Funktion	Abkürzung	Anmerkung	Seite
SIN	SI.	sin	18
COS		cos	
TAN	TA.	tan	
		Trigonometrische Funktionen	
ASN	AS.	arcsin	18
ACS	AC.	arccos	
ATN	AT.	arctan	
		Arcusfunktionen	
LN		ln	17
LOG	LO.	lg	
		Logarithmische Funktionen	
EXP	EX.	e^x Exponentialfunktion	16
$\sqrt{\ }$		$\sqrt{x}$ Quadratwurzel	16
DMS	DM.	Umrechnung Dezimalsystem - Sexagesimalsystem	17
DEG		Umrechnung Sexagesimalsystem-Dezimalsystem	17
INT		Ganzzahliger Anteil (Integer)	19
ABS	AB.	Absolutwert	20
SGN	SG.	Signum	19

2. ANWEISUNGEN

Anweisung	Abkürzung	Allgemeine Formen	Anmerkung	Seite
LET	LE.	(1) LET Zahlenvariable = Ausdruck (2) LET Zahlenvariable = Zahlenvariable (3) LET Zeichenvariable = "Text" (4) LET Zeichenvariable = Zeichenvariable	Die Zuordnungsanweisung LET darf weggelassen werden. Ausnahme: LET nach vorangehendem IF.	56
INPUT	I. IN. INP. INPU.	(1) INPUT Variable, Variable, ... (2) INPUT "Text", Variable, "Text", Variable, . . . (3) INPUT "Text"; Variable, "Text"; Variable, . . .	Eingabeanweisung zur Dateneingabe.	57
AREAD	A. AR. ARE. AREA.	AREAD Variable	Automatische Zuweisung von Daten zu Programmen, die mit Namen versehen sind und in Betriebsart DEF laufen.	59
PRINT	P. PR. PRI. PRIN.	(1) PRINT Ausdruck (2) PRINT "Text" (3) PRINT Zeichenvariable (4) PRINT { Ausdruck / "Text" / Zeichenvariable } , { Ausdruck / "Text" / Zeichenvariable } (5) PRINT { Ausdruck / "Text" / Zeichenvariable } ; { Ausdruck / "Text" / Zeichenvariable } ; . . . . . { Ausdruck / "Text" / Zeichenvariable }	Ausgabeanweisung. Programmierte und berechnete Informationen werden angezeigt.	60
PAUSE	PA. PAU. PAUS.	Die allgemeinen Formen stimmen mit denen von PRINT überein	Ausgabeanweisung entsprechend PRINT, jedoch automatische Fortführung des Programmlaufs nach 0.85 Sekunden.	63
USING	U. US. USI. USIN.	(1a) USING #···#, #···#↵ (1b) USING ENTER oder USING : (2a) { PRINT / PAUSE } USING "Format"; · · · · (2b) { PRINT / PAUSE } USING; · · · ·	Mit der Form a wird jeweils ein bestimmtes Zahlenformat festgelegt. Mit Form b wird diese Festlegung gelöscht.	63
GOTO	G. GO. GOT.	(1) GOTO Ausdruck (2) GOTO "Text" (3) GOTO Zeichenvariable	Absolute Sprunganweisung zu einer Programmzeile oder einer Programmarke.	66
IF		(1) IF Ausdruck 1 Logischer Vergleich Ausdruck 2 Befehl (2) IF Ausdruck Befehl (3) IF { "Text 1" / Zeichenvariable 1 } = { "Text 2" / Zeichenvariable 2 } Befehl (4) IF Zeichenvariable Befehl	Programmverzweigungsanweisung. Ist die Bedingung erfüllt, wird die betreffende Zeile ausgeführt. Wenn nicht, läuft das Programm mit der nächsten Zeile weiter.	67

Anweisung	Abkürzung	Allgemeine Formen	Anmerkung	Seite
THEN	T. TH. THE.		Diese Anweisung entspricht im Anschluß an eine IF-Anweisung der GOTO-Anweisung.	67
GOSUB RETURN	GOS. GOSU. RE. RET. RETU. RETUR.	GOSUB { Ausdruck / "Text" / Zeichenvariable } Unterprogramm RETURN	Sprunganweisung zum Unterprogramm, das durch Ausdruck, "Text" oder Zeichenvariable markiert ist. Der Rücksprung zur Ausgangszeile wird durch RETURN programmiert.	69
FOR TO STEP NEXT	F. FO. STE. N. NE. NEX.	(1) FOR Zahlenvariable = Ausdruck 1 TO Ausdruck 2 STEP Ausdruck 3 Schleifenprogramm NEXT Zahlenvariable (2) Wie oben aber ohne STEP Ausdruck 3 Es wird die Schrittweite 1 gewählt.	Schleifenanweisung zum mehrfachen Durchlaufen des Schleifenprogramms. Ausdruck 1: Ausgangswert der Zahlenvariablen. Ausdruck 2: Endwert der Zahlenvariablen. Ausdruck 3: Schrittweite. Mit NEXT wird der nächste Schleifendurchlauf eingeleitet.	71
STOP	S. ST. STO.	STOP	Unterbrechung des Programmlaufs.	75
END	E. EN.	END	Programmende	75
CLEAR	CL. CLE. CLEA.	CLEAR Bei manuellem Rechnen: CLEAR [ENTER]	Löschung aller Datenspeicher	75
DEGREE	DEG. DEGR. DEGRE.	DEGREE Bei manuellem Rechnen: DEGREE [ENTER]	Festlegen der Winkeleinheit Grad (Altgrad,°).	76
RADIAN	RA. RAD. RADI. RADIA.	RADIAN Bei manuellem Rechnen: RADIAN [ENTER]	Festlegen der Winkeleinheit Radiant (Bogenmaß, rad).	76
GRAD	GR. GRA.	GRAD Bei manuellem Rechnen: GRAD [ENTER]	Festlegen der Winkeleinheit Gon (Neugrad).	76
BEEP	B. BE. BEE.	BEEP Ausdruck	Akustisches Signal. Die Anzahl der Signaltöne wird durch den Ausdruck bestimmt.	76

Anweisung	Abkürzung	Allgemeine Formen	Anmerkung	Seite
REM		REM Kommentar	Kommentarvereinbarung zur Erläuterung des Programms. REM wird beim Programmlauf nicht beachtet.	76

3. KOMMANDOANWEISUNGEN

Die Kommandoanweisungen können nicht programmiert werden.

Anweisung	Abkürzung	Allgemeine Formen	Anmerkung	Seite
RUN	R. RU.	(1) RUN [ENTER] (2) RUN Ausdruck [ENTER] (3) RUN { "Text" / Zeichenvariable } [ENTER]	Befehl zur Ausführung eines Programms. Nur wirksam in den Betriebsarten RUN und DEF.	76
DEBUG	D. DE. DEB. DEBU.	Die allgemeinen Formen entsprechen denen der Anweisung RUN	Schrittweise Überprüfung von Programmen. Nur wirksam in den Betriebsarten RUN und DEF.	77
CONT	C. CO. CON.	CONT [ENTER]	Starten des Programmlaufs nach einer Programmunterbrechung. Nur wirksam in den Betriebsarten RUN und DEF.	77
LIST	L. LI. LIS.	Die allgemeinen Formen entsprechen denen der Anweisung RUN	Auflisten der Programme in Betriebsart PRO.	78
NEW		NEW [ENTER]	Vollständige Löschung von Programm und Datenregister in den Betriebsarten RUN, DEF und PRO. Vollständige Löschung des RESERVE-Registers in Betriebsart RESERVE.	79
MEM	M. ME.	MEM [ENTER]	Abfrage der verfügbaren Programmschritte oder der flexiblen Speicher.	79

Anweisung	Abkürzung	Allgemeine Formen	Anmerkung	Seite
CSAVE	CS. CSA. CSAV.	Nur manuell ausführbar: CSAVE "Blockname" ENTER	Ein Programm oder RESERVE-Ausdruck wird auf Band gespeichert.	86
CLOAD	CLO. CLOA.	Nur manuell ausführbar: CLOAD "Blockname" ENTER	Ein Programm oder RESERVE-Ausdruck wird in den Rechner geladen.	87
CLOAD?	CLO.? CLOA.?	Nur manuell ausführbar: CLOAD? "Blockname" ENTER	Die Informationen auf Band und im Rechner werden verglichen.	88
CHAIN	CH. CHA. CHAI.	(1) CHAIN "Blockname" (2) CHAIN "Blockname", Ausdruck (3) CHAIN "Blockname", { "Text" / Zeichenvariable }	Programmgesteuerter Abruf eines auf Band gespeicherten Programmblocks.	88
PRINT#	P.# PR.# PRI.# PRIN.#	(1) PRINT# "Blockname" (2) PRINT# "Blockname"; Variable	Daten werden auf Band gespeichert. Manuell und programmgesteuert einsetzbar.	90
INPUT#	I.# IN.# INP.# INPU.#	(1) INPUT# "Blockname" (2) INPUT# "Blockname"; Variable	Daten werden vom Band in den Rechner geladen. Manuell und programmgesteuert einsetzbar.	91

XIII TECHNISCHE DATEN

Stellenzahl:	Mantisse 10, Exponent 2
Logik:	Algebraische Logik mit Hierarchie
Programmiersprache:	BASIC
Kapazität:	Programmspeicher: 1424 Schritte
	Datenspeicher: Basisspeicher mit 26 Speicherplätzen Flexibler Speicher identisch mit dem Programmspeicher benötigt 8 Programmschritte pro Speicherplatz
	RESERVE-Speicher: Bis zu 18 Ausdrücke mit insgesamt 48 Schritten
	Eingabespeicher: 80 Zeichen
Pufferspeicher:	Daten: 8 Ebenen
	Funktionen: 16 Ebenen
	Klammern: Bis zu 15 Ebenen
	Unterprogramme: 4 Ebenen
	Schleifen: 4 Ebenen
Mathematische Funktionen:	Grundrechenarten, allgemeine Exponentialfunktion, Trigonometrische und Arcus-Funktionen, Logarithmische und Exponentialfunktionen, Winkelumrechnung, Quadratwurzel, Signum, Absolutwert, Integer und Vergleichsoperatoren.
Editieren:	Verschieben des Cursors nach rechts und links, Einfügen, Löschen und Überschreiben von Zeichen, Auf- und Abwärtsrollen der Programmzeilen.
Externer Speicher:	Bandgerät mit zusätzlichem Interface CE-121.
Sicherung der Permanentspeicher:	Der minimale Verbrauch der CMOS-Bausteine wird durch die Batterien gedeckt.
Anzeige:	24-stellige Punktmatrixanzeige
Bauelemente:	Höchstintegrierte CMOS-Schaltungen
Stromversorgung:	5,4V Gleichstrom
	4 Quecksilberbatterien Typ MR44 (1,35V)
	Verbrauch: Rechner 0,011 W
	Rechner mit Interface 0,013 W
Betriebszeit pro Batteriesatz:	Etwa 300 Stunden bei 20°C. Diese Zeit hängt etwas von der Batteriemarke und dem Gebrauch des Rechners ab.
Betriebstemperatur:	0°C bis 40°C
Abmessungen:	175 mm x 70 mm x 15 mm
Gewicht:	Etwa 170g
Zubehör:	Schutzhülle, Batterien, Bedienungshandbuch, Abdeckschablonen.

SHARP CORPORATION
OSAKA, JAPAN

Printed in Japan
2F5T(TINSG3636CCZZ)